

Introdução ao VHDL

Prof. Dr. Oscar Eduardo Anacona Mosquera

oscar.mosquera@ufmt.br

18 de março de 2024

Conteúdo

1 Objetivos

2 Linguagem de descrição de hardware VHDL

- Entidade de projeto
- Corpo da arquitetura
- Classes de objetos
- Tipos de dados
- Operadores
- Sobre carga de Operadores

Objetivos

Linguagem de
descrição de
hardware VHDLEntidade de projeto
Corpo da arquitetura
Classes de objetos
Tipos de dados
Operadores
Sobre carga de OperadoresDeclarações
Concorrentesselect
whenInstanciação de componentes
Declaração de processosDeclarações
sequenciaisDeclarações IF-THEN
Declaração CASE
Loop statements
Wait statements
Signal attributesVHDL and Logic
SynthesisMáquinas de Estados
Finitas (FSMs)

Bibliografia

3 Declarações Concorrentes

- select
- when
- Instanciação de componentes
- Declaração de processos

4 Declarações sequenciais

- Declarações IF-THEN
- Declaração CASE
- Loop statements
- Wait statements
- Signal attributes

5 VHDL and Logic Synthesis

6 Máquinas de Estados Finitas (FSMs)

7 Bibliografia

Objetivos

Objetivos

Linguagem de
descrição de
hardware VHDL

Entidade de projeto
Corpo da arquitetura
Classes de objetos
Tipos de dados
Operadores
Sobrecarga de Operadores

Declarações
Concorrentes

select
when
Instânciação de componentes
Declaração de processos

Declarações
sequenciais

Declarações IF-THEN
Declaração CASE
Loop statements
Wait statements
Signal attributes

VHDL and Logic
Synthesis

Máquinas de Estados
Finitas (FSMs)

Bibliografia

● Introdução à VHDL:

- ▶ Compreender os fundamentos da VHDL como uma linguagem de descrição de hardware.
- ▶ Explorar a evolução do fluxo de projeto em VHDL, desde suas origens até os métodos modernos.

● Modelo de um Arquivo VHDL: Composição e Estrutura:

- ▶ Analisar a estrutura de um arquivo VHDL, incluindo entidades, arquiteturas e bibliotecas.
- ▶ Compreender a composição de um modelo VHDL e sua importância no design de circuitos digitais.

● Declarações Concorrentes em VHDL:

- ▶ Explorar declarações concorrentes em VHDL, como processos, blocos e sinais.
- ▶ Compreender como as declarações concorrentes são usadas para descrever comportamentos simultâneos em circuitos digitais.

● Declarações Sequenciais em VHDL:

- ▶ Analisar declarações sequenciais em VHDL, como atribuições, loops e condicionais.
- ▶ Entender como as declarações sequenciais são usadas para descrever comportamentos que ocorrem em uma sequência específica.

● Máquinas de Estados Finitas em VHDL:

- ▶ Compreender os princípios fundamentais das máquinas de estados finitas (FSMs).
- ▶ Desenvolver habilidades para implementar FSMs em VHDL para sistemas sequenciais.

Conteúdo

Objetivos

Linguagem de descrição de hardware VHDL

Entidade de projeto
Corpo da arquitetura
Classes de objetos
Tipos de dados
Operadores
Sobrecarga de Operadores

Declarações Concorrentes

select
when
Instanciação de componentes
Declaração de processos

Declarações sequenciais

Declarações IF-THEN
Declaração CASE
Loop statements
Wait statements
Signal attributes

VHDL and Logic Synthesis

Máquinas de Estados Finitas (FSMs)

Bibliografia

1 Objetivos

2 Linguagem de descrição de hardware VHDL

- Entidade de projeto
- Corpo da arquitetura
- Classes de objetos
- Tipos de dados
- Operadores
- Sobrelocação de Operadores

3 Declarações Concorrentes

- select
- when
- Instanciação de componentes
- Declaração de processos

4 Declarações sequenciais

- Declarações IF-THEN
- Declaração CASE
- Loop statements
- Wait statements
- Signal attributes

5 VHDL and Logic Synthesis

6 Máquinas de Estados Finitas (FSMs)

7 Bibliografia

Objetivos

Linguagem de
descrição de
hardware VHDL

Entidade de projeto

Corpo da arquitetura

Classe de objetos

Tipos de dados

Operadores

Sobrecarga de Operadores

Declarações

Concorrentes

select

when

Instanciação de componentes

Declaração de processos

Declarações

sequenciais

Declarações IF-THEN

Declaração CASE

Loop statements

Wait statements

Signal attributes

VHDL and Logic

Synthesis

Máquinas de Estados

Finitas (FSMs)

Bibliografia

A VHDL permite que os engenheiros descrevam o comportamento e a estrutura de circuitos digitais complexos. Essa descrição pode incluir informações sobre sinais, portas, componentes, processos, temporização e outros aspectos do projeto.

- VHSIC (Very High Speed Integrated Circuit)
- Hardware
- Description
- Language

“Tell me how your circuit should behave and I will give you hardware that does the job.”

Principais características

Objetivos

Linguagem de
descrição de
hardware VHDL

Entidade de projeto

Corpo da arquitetura

Classe de objetos

Tipos de dados

Operadores

Sobrecarga de Operadores

Declarações

Concorrentes

select

when

Instanciação de componentes

Declaração de processos

Declarações

sequenciais

Declarações IF-THEN

Declaração CASE

Loop statements

Wait statements

Signal attributes

VHDL and Logic

Synthesis

Máquinas de Estados
Finitas (FSMs)

Bibliografia

- **Descrição de Hardware:** VHDL permite a descrição abstrata de sistemas digitais, abrangendo desde níveis mais altos de comportamento até detalhes de baixo nível.
- **Simulação e Verificação:** A linguagem é amplamente utilizada em simulação para verificar o comportamento e o desempenho dos projetos antes da implementação física.
- **Síntese Lógica:** VHDL é frequentemente utilizado em conjunto com ferramentas de síntese para transformar a descrição de alto nível em uma implementação específica em hardware.
- **Projeto de Circuitos Complexos:** É particularmente útil para projetar sistemas digitais complexos, como processadores, sistemas embarcados, e outros dispositivos eletrônicos.
- **Reutilização de Design:** A VHDL suporta a reutilização de design, permitindo que módulos e componentes sejam definidos e posteriormente utilizados em diferentes projetos.
- **Padrão IEEE:** VHDL é um padrão IEEE (**Institute of Electrical and Electronics Engineers**), o que contribui para sua ampla aceitação e uso na indústria.
- **Síntese para FPGAs e ASICs:** Pode ser utilizado para a síntese de projetos em FPGAs ou ASICs, convertendo a descrição em hardware físico.

Terminologia

Objetivos

Linguagem de descrição de hardware VHDL

Entidade de projeto
Corpo da arquitetura
Classes de objetos
Tipos de dados
Operadores
Sobrecarga de Operadores

Declarações Concorrentes

select
when
Instanciação de componentes
Declaração de processos

Declarações sequenciais

Declarações IF-THEN
Declaração CASE
Loop statements
Wait statements
Signal attributes

VHDL and Logic Synthesis

Máquinas de Estados Finitas (FSMs)

Bibliografia

- **HDL:** uma linguagem de descrição de hardware é uma linguagem de programação de software usada para a modelagem de um circuito hardware.
- **Modelo comportamental:** um componente é descrito pela resposta das entradas e saídas.
- **Modelo estrutural:** um componente é descrito pela interconexão de componentes/primitivas de níveis mais baixos (**abordagem top-down design**).

Modelo comportamental (Behavioral Modeling)

Objetivos

Linguagem de descrição de hardware VHDL

Entidade de projeto
Corpo da arquitetura
Classes de objetos
Tipos de dados
Operadores
Sobrecarga de Operadores

Declarações Concorrentes

select
when

Instanciação de componentes
Declaração de processos

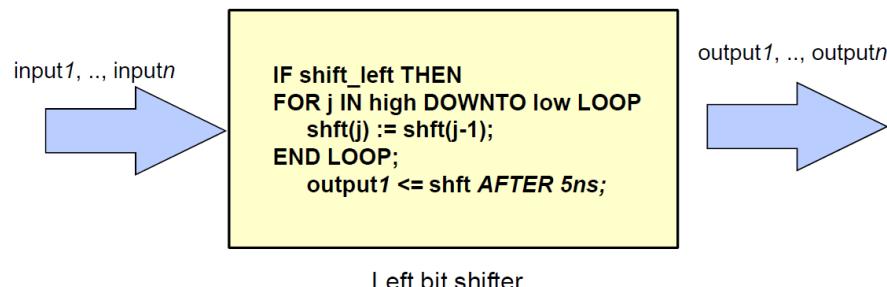
Declarações sequenciais

Declarações IF-THEN
Declaração CASE
Loop statements
Wait statements
Signal attributes

VHDL and Logic Synthesis

Máquinas de Estados Finitas (FSMs)

Bibliografia



Modelo estrutural (Structural Modeling)

Objetivos

Linguagem de
descrição de
hardware VHDL

Entidade de projeto
Corpo da arquitetura
Classes de objetos
Tipos de dados
Operadores
Sobrecarga de Operadores

Declarações
Concorrentes
select
when
Instânciação de componentes
Declaração de processos

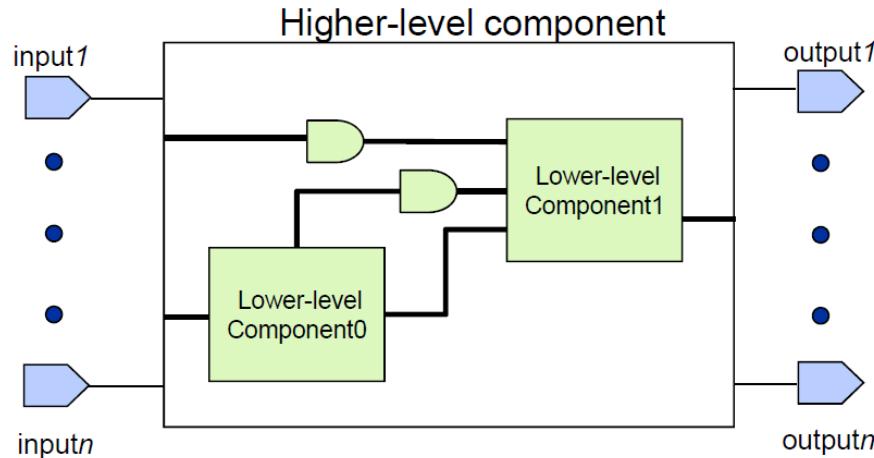
Declarações
sequenciais
Declarações IF-THEN
Declaração CASE
Loop statements
Wait statements
Signal attributes

VHDL and Logic
Synthesis

Máquinas de Estados
Finitas (FSMs)

Bibliografia

- Funcionalidade e estrutura do circuito.



Terminologia

Objetivos

Linguagem de descrição de hardware VHDL

Entidade de projeto
Corpo da arquitetura
Classes de objetos
Tipos de dados
Operadores
Sobrecarga de Operadores

Declarações Concorrentes

select
when
Instanciação de componentes
Declaração de processos

Declarações sequenciais

Declarações IF-THEN
Declaração CASE
Loop statements
Wait statements
Signal attributes

VHDL and Logic Synthesis

Máquinas de Estados Finitas (FSMs)

Bibliografia

- **Register Transfer Level (RTL):** um tipo de modelo comportamental, com o propósito de síntese. O hardware é implícito e é sintetizável.
- **Síntese:** traduzindo HDL para um circuito e, em seguida, otimizando o circuito representado.
- **Processo:** unidade básica de execução em VHDL (são convertidos para seu equivalente em hardware).

RTL Synthesis

Objetivos

Linguagem de
descrição de
hardware VHDL

Entidade de projeto
Corpo da arquitetura
Classes de objetos
Tipos de dados
Operadores
Sobrecarga de Operadores

Declarações
Concorrentes

select
when
Instânciação de componentes
Declaração de processos

Declarações
sequenciais

Declarações IF-THEN
Declaração CASE
Loop statements
Wait statements
Signal attributes

VHDL and Logic
Synthesis

Máquinas de Estados
Finitas (FSMs)

Bibliografia

PROCESS (a, b, c, d, sel)

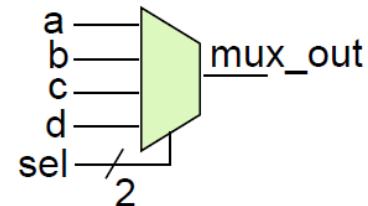
BEGIN

CASE (sel) **IS**

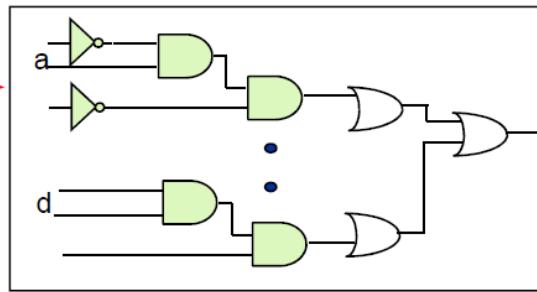
WHEN "00" => mux_out <= a;
WHEN "01" => mux_out <= b;
WHEN "10" => mux_out <= c;
WHEN "11" => mux_out <= d;

END CASE;

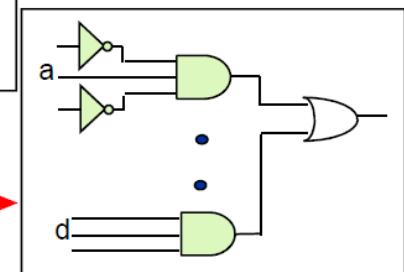
inferred



Translation



Optimization

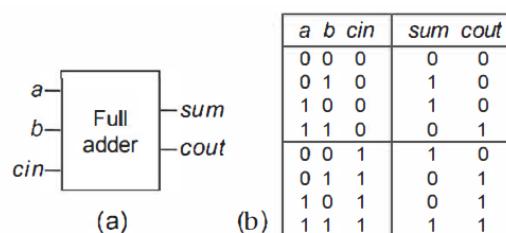


Translation of VHDL Code into a Circuit

Objetivos

Linguagem de
descrição de
hardware VHDLEntidade de projeto
Corpo da arquitetura
Classes de objetos
Tipos de dados
Operadores
Sobrecarga de OperadoresDeclarações
Concorrentesselect
when
Instanciação de componentes
Declaração de processosDeclarações
sequenciaisDeclarações IF-THEN
Declaração CASE
Loop statements
Wait statements
Signal attributesVHDL and Logic
SynthesisMáquinas de Estados
Finitas (FSMs)

Bibliografia



Output	Equation
sum	$a \oplus b \oplus cin$
$cout$	$a \cdot b + a \cdot cin + b \cdot cin$

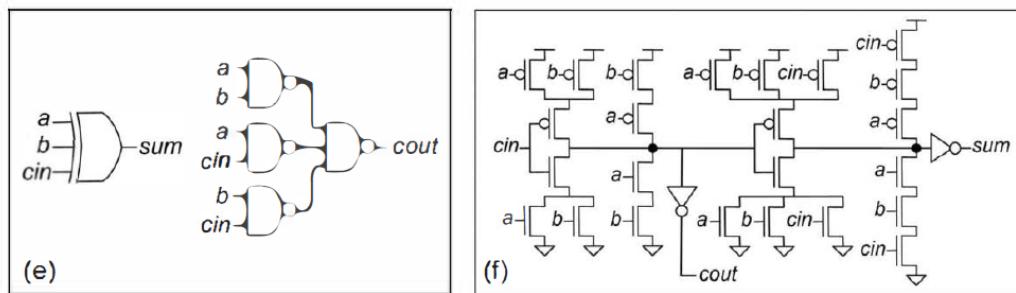
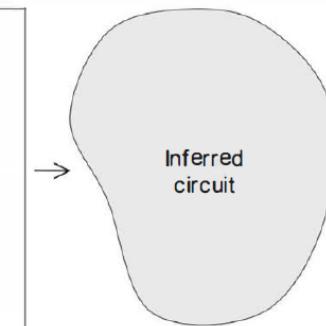
(c)

```

entity full_adder_unit is
  port (
    a, b, cin: in bit;
    sum, cout: out bit);
end entity full_adder_unit;

-----
architecture boolean_equations of full_adder_unit is
begin
  sum <= a xor b xor cin;
  cout <= (a and b) or (a and cin) or (b and cin);
end architecture boolean_equations;

```



VDHL básico

Objetivos

Linguagem de descrição de hardware VHDL

Entidade de projeto
Corpo da arquitetura
Classe de objetos
Tipos de dados
Operadores
Sobrecarga de Operadores

Declarações Concorrentes

select
when
Instanciação de componentes
Declaração de processos

Declarações sequenciais

Declarações IF-THEN
Declaração CASE
Loop statements
Wait statements
Signal attributes

VHDL and Logic Synthesis

Máquinas de Estados Finitas (FSMs)

Bibliografia

- A linguagem VHDL é composta por palavras-chave reservadas.
- A linguagem, em sua maior parte, não é sensível a maiúsculas e minúsculas.
- As declarações VHDL são terminadas com um ponto e vírgula, ";".
- VHDL é insensível a espaços em branco.
- – : **Comentário de fim de linha (EOL)**; tudo a partir do símbolo até o EOL é comentado.
- /* */ : Comentário delimitado; tudo entre os símbolos é comentado.

Palavras chave

Objetivos

Linguagem de
descrição de
hardware VHDL

Entidade de projeto
Corpo da arquitetura
Classes de objetos
Tipos de dados
Operadores
Sobrecarga de Operadores

Declarações
Concorrentes
select
when
Instanciação de componentes
Declaração de processos

Declarações
sequenciais
Declarações IF-THEN
Declaração CASE
Loop statements
Wait statements
Signal attributes

VHDL and Logic
Synthesis

Máquinas de Estados
Finitas (FSMs)

Bibliografia

abs	default	label	procedure	srl
access	disconnect	library	process	strong
after	downto	linkage	property	subtype
alias		literal	protected	
all	else	loop	pure	then
and	elsif			to
architecture	end	map	range	transport
array	entity	mod	record	type
assert	exit		register	
assume		nand	reject	unaffected
assume_guarantee	fairness	new	release	units
attribute	file	next	rem	until
	for	nor	report	use
begin	force	not	restrict	
block	function	null	restrict_guarantee	variable
body			return	vmode
buffer	generate	of	rol	vprop
bus	generic	on	ror	vunit
	group	open		
case	guarded	or	select	wait
component		others	sequence	when
configuration	if	out	severity	while
constant	impure		signal	with
context	in	package	shared	
cover	inertial	parameter	sla	xnor
	inout	port	sll	xor
	is	postponed	sra	

Unidades de Design VHDL

Objetivos

Linguagem de
descrição de
hardware VHDL

Entidade de projeto

Corpo da arquitetura

Classe de objetos

Tipos de dados

Operadores

Sobrecarga de Operadores

Declarações

Concorrentes

select

when

Instanciação de componentes

Declaração de processos

Declarações
sequenciais

Declarações IF-THEN

Declaração CASE

Loop statements

Wait statements

Signal attributes

VHDL and Logic
Synthesis

Máquinas de Estados
Finitas (FSMs)

Bibliografia

- **Entidades (Entities):** Entidades definem a interface do componente, especificando seus sinais de entrada e saída. Elas são como um contrato que descreve o comportamento externo do componente. As entidades não contêm lógica interna.
- **Arquiteturas (Architectures):** Arquiteturas definem a lógica interna de uma entidade. Elas descrevem como os sinais de entrada são manipulados para produzir os sinais de saída. Uma entidade pode ter várias arquiteturas, cada uma representando uma implementação diferente.
- **Pacotes (Packages):** Pacotes são usados para agrupar constantes, tipos de dados, subprogramas e outras declarações que podem ser compartilhadas entre várias entidades e arquiteturas. Eles são úteis para promover a reutilização e a organização do código.
- **Configurações (Configurations):** Configurações são usadas para associar uma entidade a uma arquitetura específica ou a outras entidades. Elas permitem configurar a hierarquia do projeto e selecionar as implementações desejadas para cada componente.
- **Bibliotecas (Libraries):** Bibliotecas são coleções de unidades de design relacionadas, como entidades, arquiteturas e pacotes. Elas ajudam a organizar e gerenciar o código, permitindo a reutilização de unidades em diferentes projetos.

Objetivos

Linguagem de
descrição de
hardware VHDL

Entidade de projeto
Corpo da arquitetura
Classes de objetos
Tipos de dados
Operadores
Sobrecarga de Operadores

Declarações
Concorrentes

select
when
Instanciação de componentes
Declaração de processos

Declarações
sequenciais

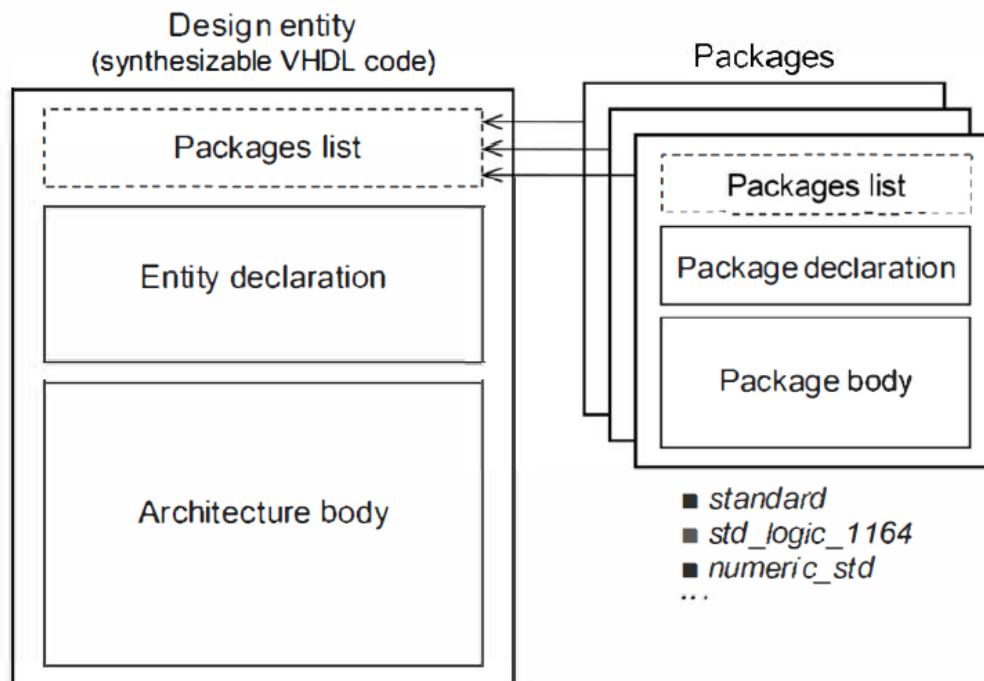
Declarações IF-THEN
Declaração CASE
Loop statements
Wait statements
Signal attributes

VHDL and Logic
Synthesis

Máquinas de Estados
Finitas (FSMs)

Bibliografia

Unidades de Design VHDL



Declaração da Entidade

Objetivos

Linguagem de
descrição de
hardware VHDL

Entidade de projeto
Corpo da arquitetura
Classes de objetos
Tipos de dados
Operadores
Sobrecarga de Operadores

Declarações
Concorrentes

select
when
Instanciação de componentes
Declaração de processos

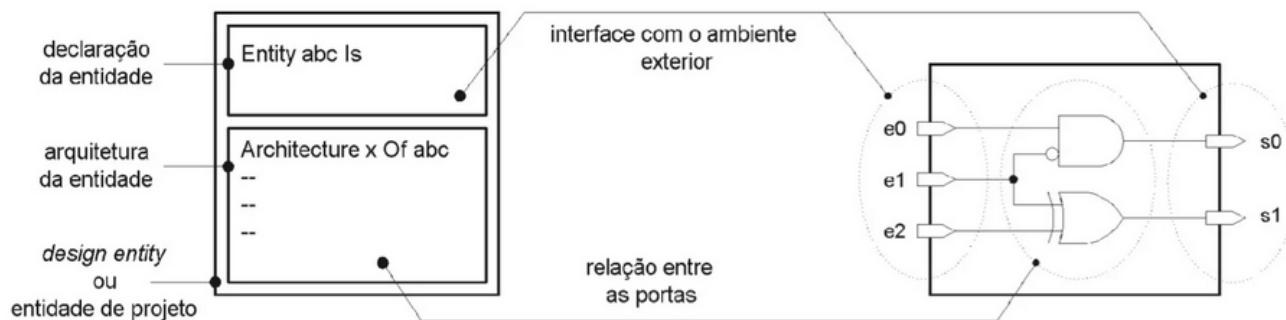
Declarações
sequenciais

Declarações IF-THEN
Declaração CASE
Loop statements
Wait statements
Signal attributes

VHDL and Logic
Synthesis

Máquinas de Estados
Finitas (FSMs)

Bibliografia



Declaração da Entidade

```
entity entity_name is

    [generic (
        constant_name: constant_type [:= constant_value];
        constant_name: constant_type [:= constant_value];
        ...);

    port (
        port_name: port_mode port_type;
        port_name: port_mode port_type;
        ...);

end [entity] [entity_name];
```


Declaração da Entidade

Objetivos

Linguagem de
descrição de
hardware VHDL

Entidade de projeto

Corpo da arquitetura

Classe de objetos

Tipos de dados

Operadores

Sobre carga de Operadores

Declarações

Concorrentes

select

when

Instanciação de componentes

Declaração de processos

Declarações sequenciais

Declarações IF-THEN

Declaração CASE

Loop statements

Wait statements

Signal attributes

VHDL and Logic Synthesis

Máquinas de Estados
Finitas (FSMs)

Bibliografia

Há quatro modos possíveis de uma porta, são eles:

- **IN:** porta de entrada.
- **OUT:** porta de saída, não pode ser referenciado internamente pela arquitetura.
- **BUFFER:** a porta opera unicamente no modo saída, pode ser referenciado internamente pela arquitetura.
- **INOUT:** caracteriza uma porta bidirecional onde uma informação pode ser apresentada ou amostrada.

Corpo da arquitetura

Objetivos

Linguagem de
descrição de
hardware VHDL

Entidade de projeto
Corpo da arquitetura

Classe de objetos

Tipos de dados

Operadores

Sobre carga de Operadores

Declarações
Concorrentes

select

when

Instanciação de componentes

Declaração de processos

Declarações
sequenciais

Declarações IF-THEN

Declaração CASE

Loop statements

Wait statements

Signal attributes

VHDL and Logic
Synthesis

Máquinas de Estados
Finitas (FSMs)

Bibliografia

A "ARCHITECTURE" em VHDL é usada para definir o comportamento ou a estrutura interna de um componente digital descrito na linguagem.

- Descreve a funcionalidade.
- Declarações são executadas concorrentemente (processos).
- Podem ser colocadas várias arquiteturas.
- **Behavioral:** mostra como funciona.
- **RTL:** A descrição dos designs é feita em termos de registradores.
- **Hybrid:** mistura de ambos estilos.

```
ARCHITECTURE nome_identificador OF entidade_abc IS
  --
  -- regiao de declaracoes:
  -- declaracoes de sinais e constantes
  -- declaracoes de componentes referenciados
  -- declaracao e corpo de subprogramas
  -- definicao de novos tipos de dados locais
  --
  BEGIN
  --
  -- comandos concorrentes
  --
END;
```

Corpo da arquitetura

Objetivos

Linguagem de
descrição de
hardware VHDL

Entidade de projeto
Corpo da arquitetura
Classes de objetos
Tipos de dados
Operadores
Sobrecarga de Operadores

Declarações
Concorrentes
select
when
Instanciação de componentes
Declaração de processos

Declarações
sequenciais
Declarações IF-THEN
Declaração CASE
Loop statements
Wait statements
Signal attributes

VHDL and Logic
Synthesis

Máquinas de Estados
Finitas (FSMs)

Bibliografia

```
architecture ... of ... is
    constant CLK_FREQ: natural := 50_000_000;
    signal data_ready: std_logic;
begin
    ...
    process (clk)
        variable count: natural range 0 to CLK_FREQ;
    begin
        ...
    end process;
    ...
end architecture;
```

Architecture body

A process

Corpo da arquitetura

Objetivos

Linguagem de
descrição de
hardware VHDL

Entidade de projeto
Corpo da arquitetura
Classes de objetos
Tipos de dados
Operadores
Sobrecarga de Operadores

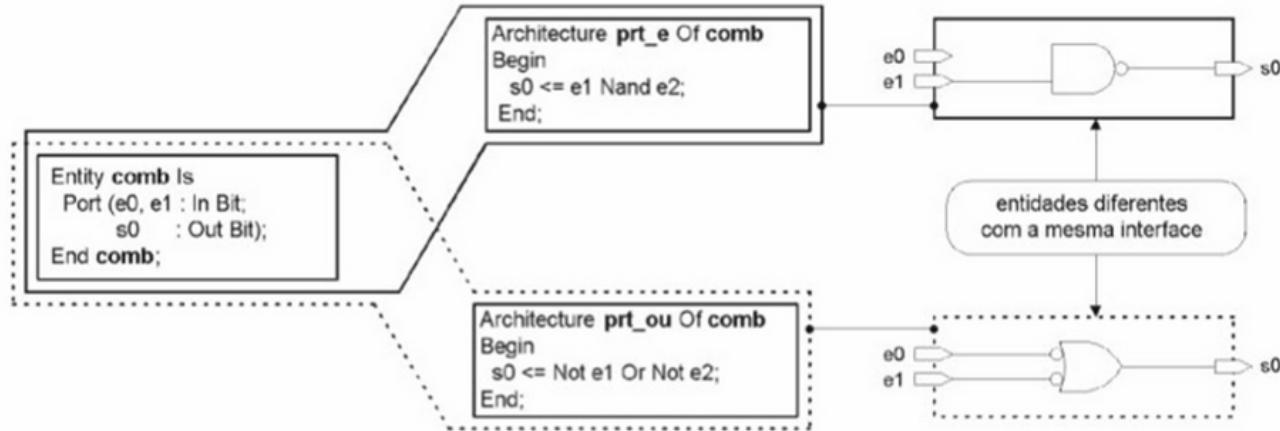
Declarações
Concorrentes
select
when
Instanciação de componentes
Declaração de processos

Declarações
sequenciais
Declarações IF-THEN
Declaração CASE
Loop statements
Wait statements
Signal attributes

VHDL and Logic
Synthesis

Máquinas de Estados
Finitas (FSMs)

Bibliografia



VHDL- Estrutura Básica de Modelagem

Objetivos

Linguagem de
descrição de
hardware VHDL

Entidade de projeto

Corpo da arquitetura

Classe de objetos

Tipos de dados

Operadores

Sobre carga de Operadores

Declarações

Concorrentes

select

when

Instanciação de componentes

Declaração de processos

Declarações
sequenciais

Declarações IF-THEN

Declaração CASE

Loop statements

Wait statements

Signal attributes

VHDL and Logic
Synthesis

Máquinas de Estados
Finitas (FSMs)

Bibliografia

```
ENTITY entity_name IS  
    generics  
    port declarations  
END ENTITY entity_name;
```

```
ARCHITECTURE arch_name OF entity_name IS  
    internal signal declarations  
    enumerated data type declarations  
    component declarations
```

```
BEGIN  
    signal assignment statements  
    PROCESS statements  
    component instantiations  
END ARCHITECTURE arch_name;
```

Classes de objetos

- **CONSTANT:** valor estático.

● **SIGNAL:** sinais são objetos que podem ter o seu valor alterado, e são empregados em regiões de código concorrente e sequencial.

- **FILE:** criação de arquivos.

Declaração de objetos das classes constante, variável e sinal:

```
-- classe    lista de nomes      tipo      valor inicial
CONSTANT nome_da_constante_a : tipo_x;                      -- constante sem valor inicial
CONSTANT nome_da_constante_a : tipo_x := valor_inicial;
CONSTANT fixo_1, fixo_2 : tipo_x := valor_inicial;

VARIABLE nome_da_variavel_c : tipo_z;                         -- variável sem valor inicial
VARIABLE nome_da_variavel_d : tipo_z := valor_inicial;        -- variável com valor inicial
VARIABLE var_1, var_2 : tipo_z := valor_inicial;              -- variáveis: mesmo tipo e valor

SIGNAL  nome_do_sinal_a : tipo_y;                            -- sinal sem valor inicial
SIGNAL  nome_do_sinal_b : tipo_y := valor_inicial;           -- sinal com valor inicial
SIGNAL  nome_x, nome_y, nome_z : tipo_y;                      -- sinais do mesmo tipo
```

Exemplos de transferência de informação entre objetos:

```
sinal_2  <= sinal_1;          -- atribuição valor para sinal
sinal_3  <= variavel_1;        -- atribuição valor para sinal
sinal_4  <= constante_1;       -- atribuição valor para sinal

variavel_2 := sinal_1;         -- atribuição valor para variável
variavel_3 := variavel_1;      -- atribuição valor para variável
variavel_4 := constante_1;     -- atribuição valor para variável
```

Classes de objetos

Objetivos

Linguagem de
descrição de
hardware VHDL

Entidade de projeto
Corpo da arquitetura
Classe de objetos
Tipos de dados
Operadores
Sobrecarga de Operadores

Declaraciones
Concorrentes

select
when
Instanciação de componentes
Declaração de processos

Declaraciones
sequenciais

Declaraciones IF-THEN
Declaração CASE
Loop statements
Wait statements
Signal attributes

VHDL and Logic
Synthesis

Máquinas de Estados
Finitas (FSMs)

Bibliografia

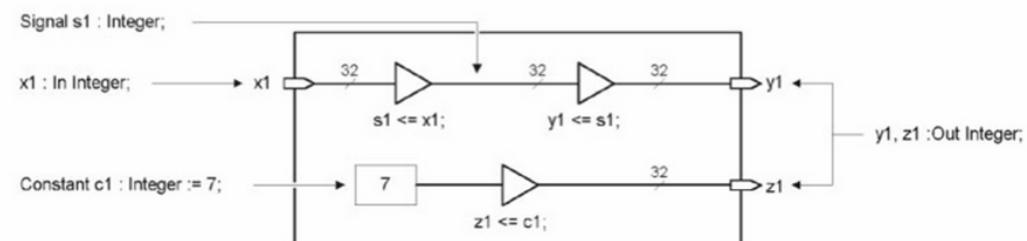
Exemplo de uma descrição completa:

```

1 ENTITY atrib_1 IS
2   PORT (x1 : IN INTEGER;
3         y1,z1 : OUT INTEGER);
4 END;
5
6 ARCHITECTURE teste OF atrib_1 IS
7   SIGNAL s1 : INTEGER;
8   CONSTANT c1 : INTEGER := 7;
9 BEGIN
10   y1 <= s1;
11   s1 <= x1;
12
13   z1 <= c1;
14 END teste;

```

Representação esquemática do código:



Classes de objetos

Objetivos

Linguagem de
descrição de
hardware VHDL

Entidade de projeto
Corpo da arquitetura
Classe de objetos
Tipos de dados
Operadores
Sobrecarga de Operadores

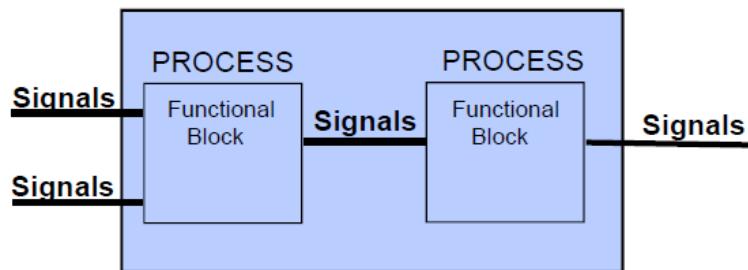
Declarações
Concorrentes
select
when
Instanciação de componentes
Declaração de processos

Declarações
sequenciais
Declarações IF-THEN
Declaração CASE
Loop statements
Wait statements
Signal attributes

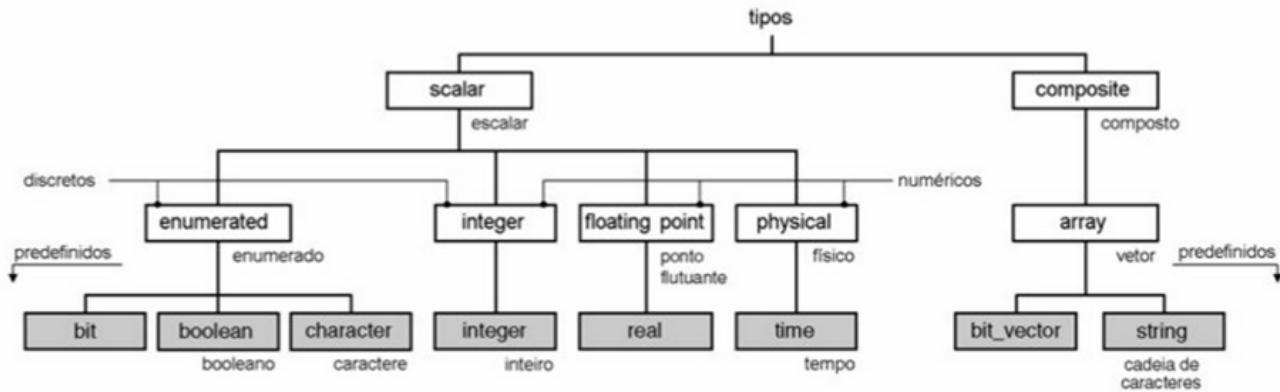
VHDL and Logic
Synthesis

Máquinas de Estados
Finitas (FSMs)

Bibliografia



Tipos de dados



Tipo predefinido	Valor	Exemplos
BIT	um, zero	1, 0
BOOLEAN	verdadeiro, falso	TRUE, FALSE
CHARACTER	caracteres ASCII	a, b, c, A, B, C, ?, <
INTEGER	$-2^{31}-1 \leq x \leq 2^{31}-1$	123, 8#173#, 16#7B#, 2#11_11_011#
NATURAL	$0 \leq x \leq 2^{31}-1$	123, 8#173#, 16#7B#
POSITIVE	$1 \leq x \leq 2^{31}-1$	2#11_11_011#
REAL	$-3.65 \times 10^{47} \leq x \leq +3.65 \times 10^{47}$	1.23, 1.23E+2, 16#7.B#E+1
TIME	$ps = 10^3 fs$ $ns = 10^3 ps$ $us = 10^3 ns$ $ms = 10^3 us$ $sec = 10^3 ms$ $min = 60 sec$ $hr = 60 min$	1 us, 100 ps, 1 fs

Objetivos

Linguagem de
descrição de
hardware VHDL

Entidade de projeto
Corpo da arquitetura
Classes de objetos
Tipos de dados
Operadores
Sobrecarga de Operadores

Declarações
Concorrentes
select
when
Instanciação de componentes
Declaração de processos

Declarações
sequenciais
Declarações IF-THEN
Declaração CASE
Loop statements
Wait statements
Signal attributes

VHDL and Logic
Synthesis

Máquinas de Estados
Finitas (FSMs)

Bibliografia

VHDL- pacotes std e IEEE

Library	Package	Typical usage
<i>std</i>	1 <i>standard</i>	For implementing all sorts of logic and integer-based arithmetic circuits but with major limitations; not recommended for arithmetic circuits and circuit ports
	2 <i>textio</i>	For dealing with text and files
	3 <i>env</i>	For communication with the simulation environment
<i>ieee</i>	4 <i>std_logic_1164</i>	For implementing any logic or arithmetic circuit (for the latter, must associate with another package, like #7, #10, or #13)
	5 <i>std_logic_textio</i>	Complement to package #2
	6 <i>numeric_bit</i>	For implementing integer arithmetic circuits with type <i>unsigned</i> or <i>signed</i> ; has <i>bit</i> as base type
	7 <i>numeric_std</i>	Same as above but with <i>std_ulogic</i> as base type
	8 <i>numeric_bit_unsigned</i>	For doing <i>unsigned</i> operations with type <i>bit_vector</i>
	9 <i>numeric_std_unsigned</i>	Same as above for type <i>std_ulogic_vector</i>
	10 <i>fixed_pkg</i>	For implementing fixed-point arithmetic circuits
	11 <i>fixed_generic_pkg</i>	
	12 <i>fixed_float_types</i>	For implementing floating-point arithmetic circuits
	13 <i>float_pkg</i>	
	14 <i>float_generic_pkg</i>	
	15 <i>math_real</i>	For determining generic parameters (support not required for synthesis, but might exist for real values that are static)

VHDL- pacotes std e IEEE: tipos sintetizáveis

Objetivos

Linguagem de descrição de hardware VHDL
Entidade de projeto

Corpo da arquitetura

Clases de objetos

Tipos de dados

Operadores

Sobre carga de Operadores

Declarações

Concorrentes

select

when

Instanciação de componentes

Declaração de processos

Declarações sequenciais

Declarações IF-THEN

Declaração CASE

Loop statements

Wait statements

Signal attributes

VHDL and Logic

Synthesis

Máquinas de Estados Finitas (FSMs)

Bibliografia

Library	Package	Type or subtype	Abbreviation	Referred to as
<i>std</i>	<i>standard</i>	bit	B	Standard types
		bit_vector	BV	
		boolean	BO	
		boolean_vector	BOV	
		integer	INT	
		integer_vector	INTV	
		natural	NAT	
		positive	POS	
		character	CHAR	
		string	STR	
<i>ieee</i>	<i>std_logic_1164</i>	std_ulogic	SU	Standard-logic types
		std_ulogic_vector	SUV	
		std_logic	SL	
		std_logic_vector	SLV	
	<i>numeric_std</i>	unsigned	UNS	Unsigned and signed types
		signed	SIG	
	<i>fixed_generic_pkg</i>	ufixed	UFIX	Fixed-point types
		sfixed	SFIX	
	<i>float_generic_pkg</i>	float	FLO	Floating-point type

VHDL- pacotes std e IEEE

Objetivos

Linguagem de
descrição de
hardware VHDL

Entidade de projeto
Corpo da arquitetura
Clases de objetos
Tipos de dados
Operadores
Sobrecarga de Operadores

Declarações
Concorrentes
select
when
Instanciação de componentes
Declaração de processos

Declarações
sequenciais
Declarações IF-THEN
Declaração CASE
Loop statements
Wait statements
Signal attributes

VHDL and Logic
Synthesis

Máquinas de Estados
Finitas (FSMs)

Bibliografia

Type STD_LOGIC

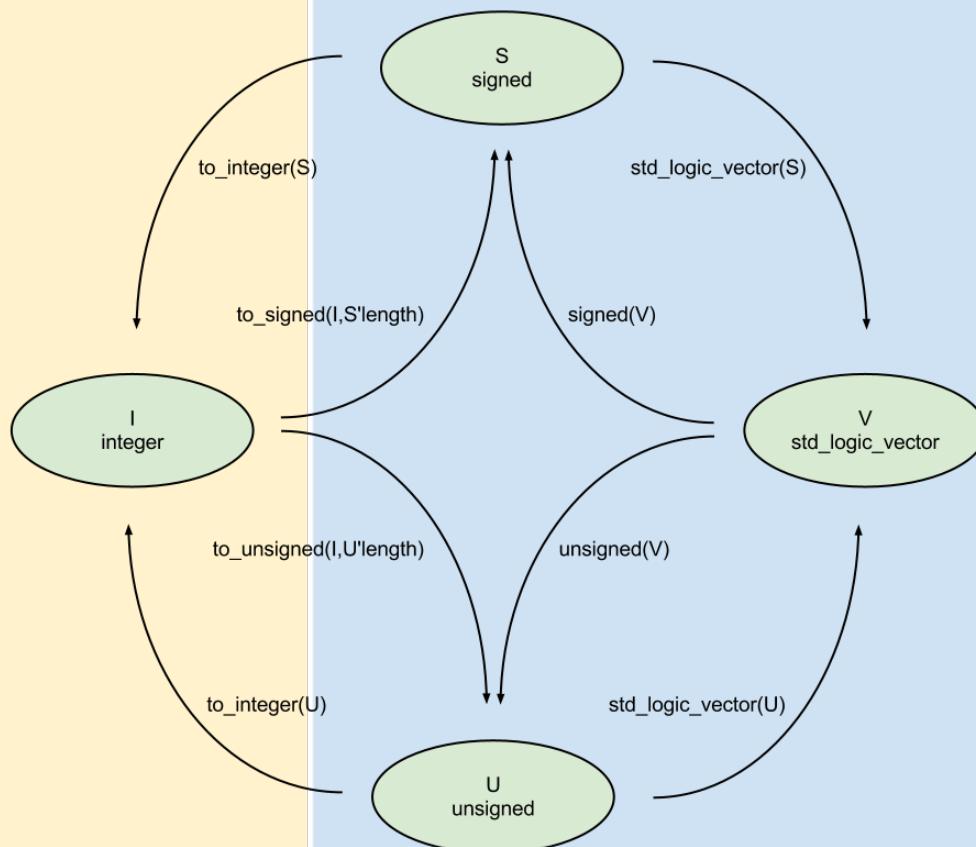
9 logic value system ('U', 'X', '0', '1', 'Z', 'W', 'L', 'H', '-')

- **1:** Logic high
- **0:** Logic low
- **X:** Unknown
- **Z:** (not 'z') Tri-state
- **-:** Don't Care
- **U:** Undefined
- **H:** Weak logic high
- **L:** Weak logic low
- **W:** Weak unknown

Conversão de tipos de dados

Numbers

Bit Vectors



Definição de novos tipos

Objetivos

Linguagem de
descrição de
hardware VHDL

Entidade de projeto

Corpo da arquitetura

Classe de objetos

Tipos de dados

Operadores

Sobrecarga de Operadores

Declarações

Concorrentes

select

when

Instanciação de componentes

Declaração de processos

Declarações
sequenciais

Declarações IF-THEN

Declaração CASE

Loop statements

Wait statements

Signal attributes

VHDL and Logic

Synthesis

Máquinas de Estados
Finitas (FSMs)

Bibliografia

A linguagem VHDL permite a criação de novos tipos enumerados, físicos e compostos.

```
TYPE estado IS (parado, inicio, caso_1, caso_2, caso_3); -- declaracao do tipo
SIGNAL abc : estado := parado; -- objeto empregando o tipo
```

Subtype

Sintetizável se o tipo base é sintetizável.

```
ARCHITECTURE logic OF subtype_test IS
  SUBTYPE word IS std_logic_vector(31 DOWNTO 0);
  SIGNAL mem_read, mem_write : word;

  SUBTYPE dec_count IS INTEGER RANGE 0 TO 9;
  SIGNAL ones, tens : dec_count;

BEGIN
```

Operadores

Objetivos

Linguagem de
descrição de
hardware VHDL

Entidade de projeto

Corpo da arquitetura

Clases de objetos

Tipos de dados

Operadores

Sobre carga de Operadores

Declarações

Concorrentes

select

when

Instanciação de componentes

Declaração de processos

Declarações
sequenciais

Declarações IF-THEN

Declaração CASE

Loop statements

Wait statements

Signal attributes

VHDL and Logic
Synthesis

Máquinas de Estados
Finitas (FSMs)

Bibliografia

Os operadores definidos são divididos em classes que estabelecem a precedência na execução das operações.

Precedência	Classe	Operadores
menor	lógicos	and or nand nor xor xnor
	relacionais	= /= < <= > >=
	deslocamento	sll srl sla sra rol ror
	adição	+ - &
	sinal	+ -
	multiplicação	* / mod rem
	diversos	** abs not

Operadores

Objetivos

Linguagem de
descrição de
hardware VHDL

Entidade de projeto

Corpo da arquitetura

Classe de objetos

Tipos de dados

Operadores

Sobre carga de Operadores

Declarações

Concorrentes

select

when

Instanciação de componentes

Declaração de processos

Declarações

sequenciais

Declarações IF-THEN

Declaração CASE

Loop statements

Wait statements

Signal attributes

VHDL and Logic

Synthesis

Máquinas de Estados
Finitas (FSMs)

Bibliografia

```

1 ENTITY int_real IS
2   PORT (cil,cil2      : OUT INTEGER RANGE 0      TO 31;
3          ci3,ci4      : OUT INTEGER RANGE 31      DOWNTO 0;
4          ci5,ci6,ci7  : OUT INTEGER RANGE -15     TO 15;
5          cr1,cr2      : OUT REAL    RANGE 0.0     TO 31.0;
6          cr3,cr4      : OUT REAL    RANGE 31.0    DOWNTO 0.0);
7 END int_real;
8
9 ARCHITECTURE teste OF int_real IS
10  CONSTANT i1 : INTEGER := 11;                      -- valor 11, base 10
11  CONSTANT i2 : INTEGER := 10#11#;                  -- valor 11, base 10
12  CONSTANT i3 : INTEGER := 2#01011#;                -- valor 11, base 2
13  CONSTANT i4 : INTEGER := 2#01_01_1#;              -- valor 11, base 2
14  CONSTANT i5 : INTEGER := 5#21#;                  -- valor 11, base 5
15  CONSTANT i6 : NATURAL := 8#13#;                 -- valor 11, base 8
16  CONSTANT i7 : POSITIVE := 16#B#;                -- valor 11, base 16
17
18  CONSTANT r1 : REAL    := 11.0;                   -- valor 11, base 10
19  CONSTANT r2 : REAL    := 1.1E01;                 -- valor 11, base 10 formato nn.nExx
20  CONSTANT r3 : REAL    := 2#01011.0#;             -- valor 11, base 2
21  CONSTANT r4 : REAL    := 8#1.3#E01;              -- valor 11, base 8 formato nn.nExx
22  CONSTANT r5 : REAL    := 16#B.0#;                -- valor 11, base 16
23 BEGIN
24   cil <= i1; ci2 <= i2; ci3 <= i3; ci4 <= i4; ci5 <= i5; ci6 <= i6; ci7 <= i7;
25   cr1 <= r1; cr2 <= r2; cr3 <= r3; cr4 <= r5;
26 END teste;

```

Operadores

Objetivos

Linguagem de
descrição de
hardware VHDL

Entidade de projeto

Corpo da arquitetura

Clases de objetos

Tipos de dados

Operadores

Sobreulação de Operadores

Declarações

Concorrentes

select

when

Instanciação de componentes

Declaração de processos

Declarações

sequenciais

Declarações IF-THEN

Declaração CASE

Loop statements

Wait statements

Signal attributes

VHDL and Logic

Synthesis

Máquinas de Estados

Finitas (FSMs)

Bibliografia

```

1 ENTITY std_xal IS
2   PORT( a, b, c, d           : IN BIT;
3         x1, x2, x3, x4, x5 : OUT BIT);
4 END std_xal;
5
6 ARCHITECTURE exemplo OF std_xal IS
7 BEGIN
8   x1 <= a OR NOT b;          -- Certo: operador NOT tem precedencia mais elevada
9   x2 <= a AND b AND c;      -- Certo: operadores iguais
10  -- x3 <= a AND b OR c;     -- Errado: expressao ambigua x3=(a.b)+c ou x3=a.(b+c) ?
11  x3 <=(a AND b) OR c;      -- Certo: empregando parenteses
12  -- x4 <= a AND b OR c AND d; -- Errado: expressao ambigua, operadores com mesma precedencia
13  x4 <=(a AND b) OR (c AND d); -- Certo: x4 = a.b + c.d
14  -- x5 <= a NAND b NAND c;  -- Errado: operadores com negacao necessitam parenteses
15  x5 <=(a NAND b) NAND c;    -- Certo: operador com negacao entre parenteses
16 END exemplo;

```

Operadores

Objetivos

Linguagem de
descrição de
hardware VHDL

Entidade de projeto

Corpo da arquitetura

Classes de objetos

Tipos de dados

Operadores

Sobreulação de Operadores

Declarações

Concorrentes

select

when

Instanciação de componentes

Declaração de processos

Declarações

sequenciais

Declarações IF-THEN

Declaração CASE

Loop statements

Wait statements

Signal attributes

VHDL and Logic

Synthesis

Máquinas de Estados
Finitas (FSMs)

Bibliografia

```
1 -----  
2 library ieee;  
3 use ieee.std_logic_1164.all;  
4  
5 entity parity_detector is  
6     port (  
7         byte: in std_logic_vector(7 downto 0);  
8         parity: out std_logic);  
9     end entity;  
10  
11 architecture tree_type of parity_detector is  
12 begin  
13     parity <= ((byte(0) xor byte(1)) xor (byte(2) xor byte(3))) xor  
14             ((byte(4) xor byte(5)) xor (byte(6) xor byte(7)));  
15 end architecture;  
16 -----
```

Operadores: sinais usadas como interligação

Objetivos

Linguagem de
descrição de
hardware VHDL

Entidade de projeto
Corpo da arquitetura
Classes de objetos
Tipos de dados
Operadores
Sobreulação de Operadores

Declarações
Concorrentes
select
when
Instanciação de componentes
Declaração de processos

Declarações
sequenciais
Declarações IF-THEN
Declaração CASE
Loop statements
Wait statements
Signal attributes

VHDL and Logic
Synthesis

Máquinas de Estados
Finitas (FSMs)

Bibliografia

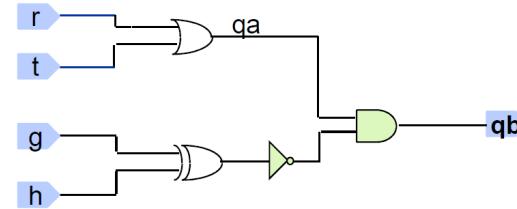
```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
```

```
ENTITY simp IS
  PORT (
    r, t, g, h : IN STD_LOGIC;
    qb : OUT STD_LOGIC
  );
END ENTITY simp;
```

```
ARCHITECTURE logic OF simp IS
  SIGNAL qa : STD_LOGIC; ←
BEGIN
```

```
    qa <= r OR t;
    qb <= (qa AND NOT (g XOR h));
```

```
END ARCHITECTURE logic;
```



*Signal declaration
inside architecture*

- r, t, g, h, and qb are signals (by default)
- qa is a buried signal and needs to be declared

Operadores: função aritmética

Objetivos

Linguagem de
descrição de
hardware VHDL

Entidade de projeto
Corpo da arquitetura
Classe de objetos
Tipos de dados
Operadores
Sobrecarga de Operadores

Declarações
Concorrentes
select
when
Instanciação de componentes
Declaração de processos

Declarações
sequenciais
Declarações IF-THEN
Declaração CASE
Loop statements
Wait statements
Signal attributes

VHDL and Logic
Synthesis

Máquinas de Estados
Finitas (FSMs)

Bibliografia

```
ENTITY opr IS
  PORT (
    a : IN INTEGER RANGE 0 TO 16;
    b : IN INTEGER RANGE 0 TO 16;
    sum : OUT INTEGER RANGE 0 TO 32
  );
END ENTITY opr;

ARCHITECTURE example OF opr IS
BEGIN
  sum <= a + b;
END ARCHITECTURE example;
```

*The VHDL compiler can
understand this operation
because an arithmetic
operation is defined for the
built-in data type
INTEGER*

Função/Pacote de Sobrecarga de Operadores

Objetivos

Linguagem de
descrição de
hardware VHDL

Entidade de projeto
Corpo da arquitetura
Classes de objetos
Tipos de dados
Operadores
Sobrecarga de Operadores

Declarações
Concorrentes
select
when
Instanciação de componentes
Declaração de processos

Declarações
sequenciais
Declarações IF-THEN
Declaração CASE
Loop statements
Wait statements
Signal attributes

VHDL and Logic
Synthesis

Máquinas de Estados
Finitas (FSMs)

Bibliografia

Pacotes que definem funções de sobrecarga de operadores podem ser encontrados na BIBLIOTECA IEEE.

- **STD_LOGIC_ARITH**: arithmetic functions
- **STD_LOGIC_SIGNED**: signed arithmetic functions
- **STD_LOGIC_UNSIGNED**: unsigned arithmetic functions
- **NUMERIC_STD**: signed and unsigned arithmetic

Por exemplo, o pacote **STD_LOGIC_UNSIGNED** define algumas das seguintes funções:

```

FUNCTION "+" (l: STD_LOGIC_VECTOR; r: STD_LOGIC_VECTOR) RETURN STD_LOGIC_VECTOR;
FUNCTION "+" (l: STD_LOGIC_VECTOR; r: INTEGER) RETURN STD_LOGIC_VECTOR;
FUNCTION "+" (l: INTEGER; r: std_logic_vector) RETURN STD_LOGIC_VECTOR;
FUNCTION "+" (l: STD_LOGIC_VECTOR; r: STD_LOGIC) RETURN STD_LOGIC_VECTOR;
FUNCTION "+" (l: STD_LOGIC; r: STD_LOGIC_VECTOR) RETURN STD_LOGIC_VECTOR;

FUNCTION "-" (l: STD_LOGIC_VECTOR; r: STD_LOGIC_VECTOR) RETURN STD_LOGIC_VECTOR;
FUNCTION "-" (l: STD_LOGIC_VECTOR; r: INTEGER) RETURN STD_LOGIC_VECTOR;
FUNCTION "-" (l: INTEGER; r: STD_LOGIC_VECTOR) RETURN STD_LOGIC_VECTOR;
FUNCTION "-" (l: STD_LOGIC_VECTOR; r: STD_LOGIC) RETURN STD_LOGIC_VECTOR;
FUNCTION "-" (l: STD_LOGIC; r: STD_LOGIC_VECTOR) RETURN STD_LOGIC_VECTOR;
```

Função/Pacote de Sobrecarga de Operadores

Objetivos

Linguagem de
descrição de
hardware VHDL

Entidade de projeto

Corpo da arquitetura

Classes de objetos

Tipos de dados

Operadores

Sobrecarga de Operadores

Declarações

Concorrentes

select
when

Instanciação de componentes

Declaração de processos

Declarações
sequenciais

Declarações IF-THEN

Declaração CASE

Loop statements

Wait statements

Signal attributes

VHDL and Logic
Synthesis

Máquinas de Estados
Finitas (FSMs)

Bibliografia

```
LIBRARY IEEE;  
USE IEEE.STD_LOGIC_1164.ALL;  
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
```

```
ENTITY overload IS  
  PORT (  
    a : IN STD_LOGIC_VECTOR (4 DOWNTO 0);  
    b : IN STD_LOGIC_VECTOR (4 DOWNTO 0);  
    sum : OUT STD_LOGIC_VECTOR (4 DOWNTO 0)  
  );  
END ENTITY overload;
```

```
ARCHITECTURE example OF overload IS  
BEGIN  
  sum <= a + b;  
END ARCHITECTURE example;
```

*Include these statements
at the beginning of a
design file*

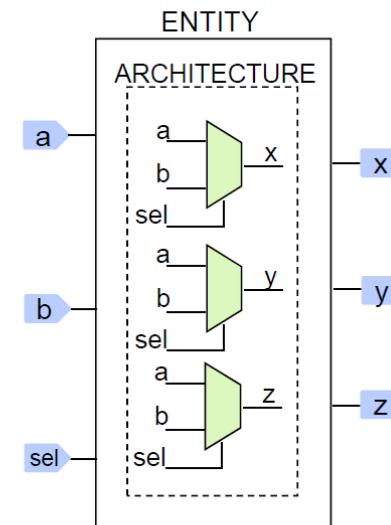
*This allows us to
perform arithmetic on
non-built-in data types*

Exemplo

```

ENTITY cmpl_sig IS
    PORT (
        a, b, sel      : IN  BIT;
        x, y, z      : OUT BIT
    );
END ENTITY cmpl_sig;
ARCHITECTURE logic OF cmpl_sig IS
BEGIN
    -- simple signal assignment
    x <= (a AND NOT sel) OR (b AND sel);
    -- conditional signal assignment
    y <= a WHEN sel='0' ELSE
        b;
    -- selected signal assignment
    WITH sel SELECT
        z <= a WHEN '0',
            b WHEN '1',
            '0' WHEN OTHERS;
END ARCHITECTURE logic;
CONFIGURATION cmpl_sig_conf OF cmpl_sig IS
    FOR logic
        END FOR;
END CONFIGURATION cmpl_sig_conf;

```



Conteúdo

1 Objetivos

2 Linguagem de descrição de hardware VHDL

- Entidade de projeto
- Corpo da arquitetura
- Classes de objetos
- Tipos de dados
- Operadores
- Sobre carga de Operadores

Objetivos

Linguagem de
descrição de
hardware VHDLEntidade de projeto
Corpo da arquitetura
Classes de objetos
Tipos de dados
Operadores
Sobre carga de OperadoresDeclarações
Concorrentesselect
when
Instanciação de componentes
Declaração de processosDeclarações
sequenciaisDeclarações IF-THEN
Declaração CASE
Loop statements
Wait statements
Signal attributesVHDL and Logic
SynthesisMáquinas de Estados
Finitas (FSMs)

Bibliografia

3 Declarações Concorrentes

- select
- when
- Instanciação de componentes
- Declaração de processos

4 Declarações sequenciais

- Declarações IF-THEN
- Declaração CASE
- Loop statements
- Wait statements
- Signal attributes

5 VHDL and Logic Synthesis

6 Máquinas de Estados Finitas (FSMs)

7 Bibliografia

Objetivos

Linguagem de
descrição de
hardware VHDLEntidade de projeto
Corpo da arquitetura
Classes de objetos
Tipos de dados
Operadores
Sobrecarga de OperadoresDeclarações
Concorrentesselect
when
Instanciação de componentes
Declaração de processosDeclarações
sequenciaisDeclarações IF-THEN
Declaração CASE
Loop statements
Wait statements
Signal attributesVHDL and Logic
SynthesisMáquinas de Estados
Finitas (FSMs)

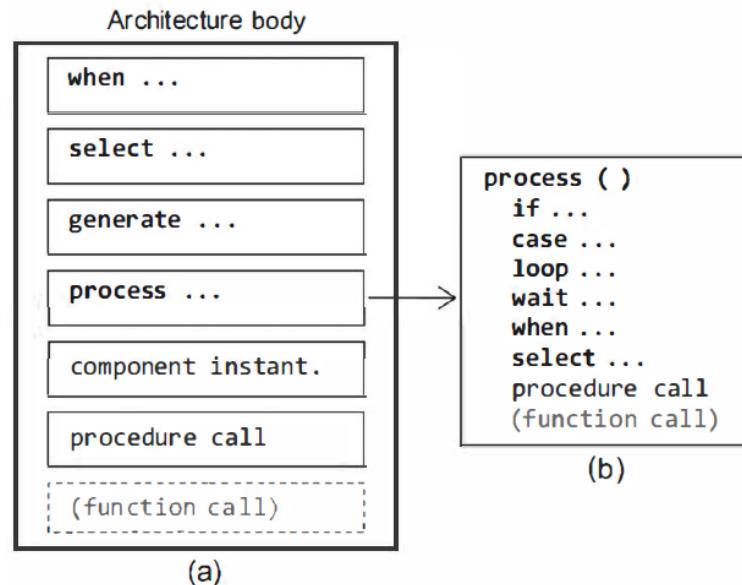
Bibliografia

Declarações Concorrentes

- Declaração **when**
- Declaração **select**
- Declaração **generate**
- Declaração **Process**

Declarações concorrentes

VHDL é inherentemente concorrente em vez de sequencial.



Declaração select

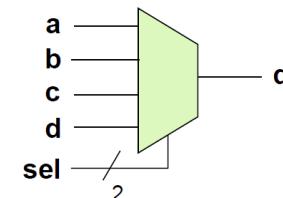
- Todas as possíveis condições devem ser consideradas.
- **when others** avalia todas as outras condições possíveis que não são especificamente declaradas.

■ Format:

```
WITH <expression> SELECT
    <signal_name> <= <signal/value> WHEN <condition_1>,
    <signal/value> WHEN <condition_2>,
    ...
    <signal/value> WHEN OTHERS;
```

■ Example:

```
WITH sel SELECT
    q <= a WHEN "00",
    b WHEN "01",
    c WHEN "10",
    d WHEN OTHERS;
```



Implied process

Declaração select

Objetivos

Linguagem de descrição de hardware VHDL

- Entidade de projeto
- Corpo da arquitetura
- Classe de objetos
- Tipos de dados
- Operadores
- Sobrecarga de Operadores

Declarações Concorrentes

- select
- when
- Instanciação de componentes
- Declaração de processos

Declarações sequenciais

- Declarações IF-THEN
- Declaração CASE
- Loop statements
- Wait statements
- Signal attributes

VHDL and Logic Synthesis

Máquinas de Estados Finitas (FSMs)

Bibliografia

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
```

```
ENTITY cmpl_sig IS
```

```
    PORT (
        a, b, sel : IN STD_LOGIC;
        z : OUT STD_LOGIC
    );
```

```
END ENTITY cmpl_sig;
```

```
ARCHITECTURE logic OF cmpl_sig IS
BEGIN
```

-- Selected signal assignment

```
    WITH sel SELECT
```

```
        z <= a WHEN '0',
        b WHEN '1',
        '0' WHEN OTHERS;
```

```
END ARCHITECTURE logic;
```

sel is of **STD_LOGIC** data type

- What are the values for a **STD_LOGIC** data type
- Answer: {‘0’, ‘1’, ‘X’, ‘Z’...}
- Therefore, is the **WHEN OTHERS** clause necessary?
- Answer: YES

Declaração select

Objetivos

Linguagem de descrição de hardware VHDL

Entidade de projeto
Corpo da arquitetura
Classes de objetos
Tipos de dados
Operadores
Sobrecarga de Operadores

Declarções
Concorrentes
select
when
Instanciação de componentes
Declaração de processos

Declarções sequenciais
Declarções IF-THEN
Declaração CASE
Loop statements
Wait statements
Signal attributes

VHDL and Logic
Synthesis

Máquinas de Estados Finitas (FSMs)

Bibliografia

```

LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

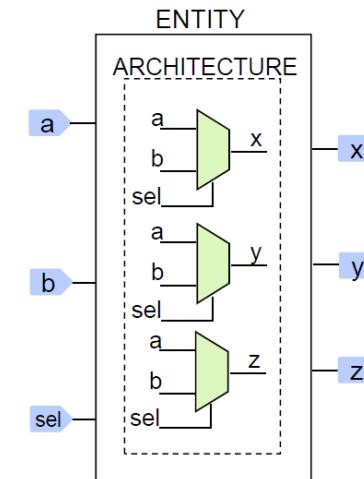
ENTITY cmpl_sig IS
  PORT (
    a, b, sel : IN STD_LOGIC;
    x, y, z : OUT STD_LOGIC
  );
END ENTITY cmpl_sig;

ARCHITECTURE logic OF cmpl_sig IS
BEGIN
  -- Simple signal assignment
  x <= (a AND NOT sel) OR (b AND sel);

  -- Conditional signal assignment
  y <= a WHEN sel='0' ELSE
    b;
  -- Selected signal assignment
  WITH sel SELECT
    z <= a WHEN '0',
    b WHEN '1',
    X' WHEN OTHERS;
END ARCHITECTURE logic;

```

- The signal assignments execute in parallel, and therefore the order we list the statements should not affect the outcome



Declaração when

Objetivos

Linguagem de
descrição de
hardware VHDL

Entidade de projeto
Corpo da arquitetura
Classes de objetos
Tipos de dados
Operadores
Sobrecarga de Operadores

Declarações
Concorrentes

select
when
Instanciação de componentes
Declaração de processos

Declarações
sequenciais

Declarações IF-THEN
Declaração CASE
Loop statements
Wait statements
Signal attributes

VHDL and Logic
Synthesis

Máquinas de Estados
Finitas (FSMs)

Bibliografia

```
target <= value when condition else
            value when condition else
            value;
```

```
target <= value when condition else
            value when condition else
            value when condition;
```

Ending in *when condition*:

```
y <= a when sel=0 else
      b when sel=1 else
      c when sel=2 else
      d when sel=3;
```

Ending in *else value*:

```
y <= a when sel=0 else
      b when sel=1 else
      c when sel=2 else
      d;
```

Declaração when

Objetivos

Linguagem de
descrição de
hardware VHDL

Entidade de projeto
Corpo da arquitetura
Classes de objetos
Tipos de dados
Operadores
Sobrecarga de Operadores

Declaraciones
Concorrentes

select
when
Instanciação de componentes
Declaração de processos

Declaraciones
sequenciais

Declaraciones IF-THEN
Declaração CASE
Loop statements
Wait statements
Signal attributes

VHDL and Logic
Synthesis

Máquinas de Estados
Finitas (FSMs)

Bibliografia

```
1  -----
2  library ieee;
3  use ieee.std_logic_1164.all;
4
5  entity priority_encoder is
6      port (
7          inp: in std_logic_vector(3 downto 0);
8          outp: out std_logic_vector(3 downto 0));
9  end entity;
10
11 architecture lut of priority_encoder is
12 begin
13     with inp select?
14         outp <= "1000" when "1--",
15                     "0100" when "01--",
16                     "0010" when "001-",
17                     "0001" when "0001",
18                     "0000" when others;
19 end architecture;
20 -----
```

Instanciação de componentes

Objetivos

Linguagem de
descrição de
hardware VHDL

Entidade de projeto
Corpo da arquitetura
Classes de objetos
Tipos de dados
Operadores
Sobrecarga de Operadores

Declarações
Concorrentes

select
when
Instanciação de componentes
Declaração de processos

Declarações
sequenciais
Declarações IF-THEN
Declaração CASE
Loop statements
Wait statements
Signal attributes

VHDL and Logic
Synthesis

Máquinas de Estados
Finitas (FSMs)

Bibliografia

Refere-se ao processo de criar uma instância ou uma cópia de uma entidade (design ou componente) dentro de um design maior. Isso permite reutilizar funcionalidades já definidas em um novo contexto, evitando a necessidade de reescrever o código.

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
```

Upper-level of hierarchy design must have
a component declaration for a lower-
level design before it can be instantiated

```
ENTITY tolleab IS
  PORT (
    tclk, tcross, tnickel, tdime, tquarter : IN STD_LOGIC;
    tgreen, tred : OUT STD_LOGIC
  );
END ENTITY tolleab;
```

Component declaration

```
ARCHITECTURE tolleab_arch OF tolleab IS
```

```
  COMPONENT tolly
    PORT(
      clk, cross, nickel, dime, quarter : IN STD_LOGIC;
      green, red : OUT STD_LOGIC;
    );
  END COMPONENT;
```

Lower-level port

```
BEGIN
```

```
  U1 : tolly PORT MAP (clk => tclk, cross => tcross,
                        nickel => tnickel, dime => tdime, quarter => tquarter,
                        green => tgreen, red => tred);
END ARCHITECTURE tolleab_arch;
```

Dime => tdime

Instance label/name

Current-level port

Named Association

Component instantiation

Instanciação de componentes

Objetivos

Linguagem de
descrição de
hardware VHDL

Entidade de projeto
Corpo da arquitetura
Classes de objetos
Tipos de dados
Operadores
Sobrecarga de Operadores

Declarações
Concorrentes

select
when

Instanciação de componentes
Declaração de processos

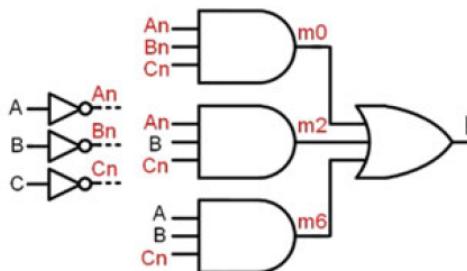
Declarações
sequenciais

Declarações IF-THEN
Declaração CASE
Loop statements
Wait statements
Signal attributes

VHDL and Logic
Synthesis

Máquinas de Estados
Finitas (FSMs)

Bibliografia



```
entity INV1 is
  port (A : in bit;
        F : out bit);
end entity;

entity AND3 is
  port (A,B,C : in bit;
        F      : out bit);
end entity;

entity OR3 is
  port (A,B,C : in bit;
        F      : out bit);
end entity;
```

```
entity SystemX is
  port (A, B, C : in bit;
        F         : out bit);
end entity;

architecture SystemX_arch of SystemX is
  signal An, Bn, Cn : bit; -- declare signals
  signal m0, m2, m6 : bit; -- declare signals

  component INV1           -- declare INV1
    port (A : in bit;
          F : out bit);
  end component;

  component AND3           -- declare AND3
    port (A,B,C : in bit;
          F      : out bit);
  end component;

  component OR3            -- declare OR3
    port (A,B,C : in bit;
          F      : out bit);
  end component;

begin
  U1 : INV1 port map (A=>An, F=>An);
  U2 : INV1 port map (A=>Bn, F=>Bn);
  U3 : INV1 port map (A=>Cn, F=>Cn);

  U4 : AND3 port map (A=>An, B=>Bn, C=>Cn, F=>m0);
  U5 : AND3 port map (A=>A, B=>B, C=>Cn, F=>m2);
  U6 : AND3 port map (A=>A, B=>B, C=>Cn, F=>m6);

  U7 : OR3 port map (A=>m0, B=>m2, C=>m6, F=>F);
end architecture;
```

The entity is named SystemX.

Internal signals are needed to connect the sub-systems.

The three lower level sub-systems are declared as components in SystemX.

The components are instantiated and connected using explicit port mapping in order to describe the behavior of the logic diagram.

NOTs

AND's

OR

Declaração de processos

Objetivos

Linguagem de
descrição de
hardware VHDL

Entidade de projeto
Corpo da arquitetura
Classes de objetos
Tipos de dados
Operadores
Sobrecarga de Operadores

Declaraciones
Concorrentes

select
when

Instanciação de componentes

Declaração de processos

Declaraciones
sequenciais

Declaraciones IF-THEN
Declaração CASE
Loop statements
Wait statements
Signal attributes

VHDL and Logic
Synthesis

Máquinas de Estados
Finitas (FSMs)

Bibliografia

Process()

O VHDL usa um processo para modelar atribuições de sinal baseadas em um evento. Dentro de um processo em VHDL, as declarações são executadas sequencialmente. Isso significa que as instruções dentro do bloco de processo são executadas em ordem, uma após a outra, seguindo a ordem em que foram escritas no código.

Sensitivity list

Uma lista de sensibilidade é um mecanismo para controlar quando um processo é acionado (ou iniciado). Uma lista de sensibilidade contém uma lista de sinais aos quais o processo é sensível. Se houver uma transição em qualquer um dos sinais da lista, o processo será acionado e as atribuições de sinal no processo serão feitas.

```
process_name : process (<signal_name1>, <signal_name2>, ...)

-- variable declarations

begin

    sequential_signal_assignment_1
    sequential_signal_assignment_2
    :

end process;

FlipFlop : process (Clock)
begin
    Q <= D;
end process;
```

Objetivos

Linguagem de
descrição de
hardware VHDLEntidade de projeto
Corpo da arquitetura
Classes de objetos
Tipos de dados
Operadores
Sobrecarga de OperadoresDeclarações
Concorrentesselect
when
Instanciação de componentes
Declaração de processosDeclarações
sequenciaisDeclarações IF-THEN
Declaração CASE
Loop statements
Wait statements
Signal attributesVHDL and Logic
SynthesisMáquinas de Estados
Finitas (FSMs)

Bibliografia

Declaração de processos

Example: Behavior of Sequential Signal Assignments within a Process

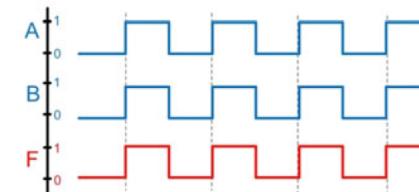
For the following system:



```
entity Ex is
  port (A : in bit;
        F : out bit);
end entity;
```

The output F will match the input A when modeled with the following process:

```
architecture Ex_arch of Ex is
  signal B : bit;
begin
  Proc_Ex : process (A)
  begin
    B <= A;
    F <= not B;
  end process;
end architecture;
```



Example: Behavior of Concurrent Signal Assignments outside a Process

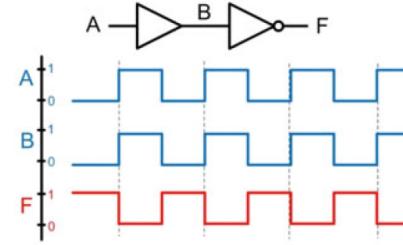
For the following system:



```
entity Ex is
  port (A : in bit;
        F : out bit);
end entity;
```

The output F will be the complement of input A when the assignments are executed concurrently.

```
architecture Ex_arch of Ex is
  signal B : bit;
begin
  B <= A;
  F <= not B;
end architecture;
```



Declaração de processos

Objetivos

Linguagem de
descrição de
hardware VHDL

Entidade de projeto
Corpo da arquitetura
Classes de objetos
Tipos de dados
Operadores
Sobrecarga de Operadores

Declarações
Concorrentes

select
when
Instanciação de componentes
Declaração de processos

Declarações
sequenciais

Declarações IF-THEN
Declaração CASE
Loop statements
Wait statements
Signal attributes

VHDL and Logic
Synthesis

Máquinas de Estados
Finitas (FSMs)

Bibliografia

Example: Behavior of Variable Assignments within a Process

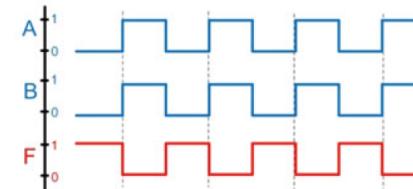
For the following system:



```
entity Ex is
  port (A : in bit;
        F : out bit);
end entity;
```

```
architecture Ex_arch of Ex is
  signal B : bit;
begin
  Proc_Ex : process (A)
    variable temp : bit := '0';
  begin
    temp := A;
    B   <= temp;
    F   <= not temp;
  end process;
end architecture;
```

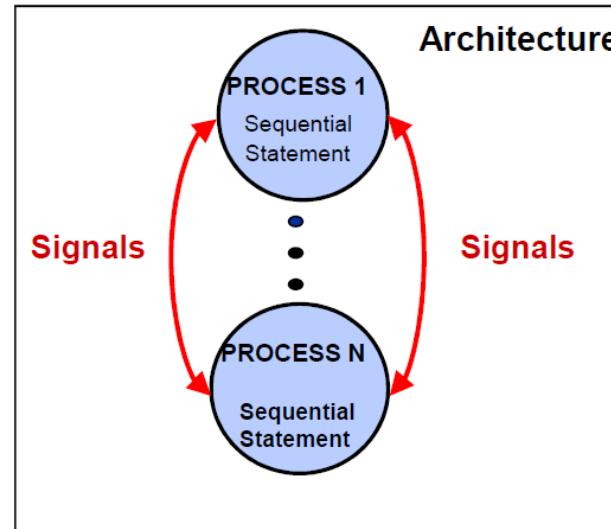
The output F will match the input A when modeled with the following process:



Declaração de processos

Processos concorrentes

- Uma arquitetura pode ter múltiplos processos concorrentes.
- Cada processo é executado em paralelo com outros processos.
- Sim, a ordem das declarações dentro de um processo em VHDL importa significativamente. A ordem das declarações determina a sequência de operações executadas pelo processo e pode afetar o comportamento do circuito descrito pelo código VHDL.



Declaração de processos

Objetivos

Linguagem de
descrição de
hardware VHDL

Entidade de projeto
Corpo da arquitetura
Classes de objetos
Tipos de dados
Operadores
Sobrecarga de Operadores

Declarções
Concorrentes
select
when
Instanciação de componentes
Declaração de processos

Declarções
sequenciais
Declarções IF-THEN
Declaração CASE
Loop statements
Wait statements
Signal attributes

VHDL and Logic
Synthesis

Máquinas de Estados
Finitas (FSMs)

Bibliografia

Equivalent Functions?? YES

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY simp IS
  PORT (
    a, b : IN STD_LOGIC;
    y : OUT STD_LOGIC
  );
END ENTITY simp;

ARCHITECTURE logic OF simp IS
  SIGNAL c : STD_LOGIC;
BEGIN
  c <= a AND b; ←
  y <= c; ←
END ARCHITECTURE logic;
```

c AND y get executed and updated in parallel at the end of the process within one simulation cycle

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
```

```
ENTITY simp_prc IS
  PORT (
    a,b : IN STD_LOGIC;
    y : OUT STD_LOGIC
  );
END ENTITY simp_prc;
```

```
ARCHITECTURE logic OF simp_prc IS
  SIGNAL c : STD_LOGIC;
BEGIN
  process1: PROCESS (a, b)
  BEGIN
    c <= a AND b;
  END PROCESS process1;
  process2: PROCESS (c)
  BEGIN
    y <= c;
  END PROCESS process2;
END ARCHITECTURE logic;
```



Declaração de processos

Objetivos

Linguagem de
descrição de
hardware VHDL

Entidade de projeto
Corpo da arquitetura
Classes de objetos
Tipos de dados
Operadores
Sobrecarga de Operadores

Declarações
Concorrentes
select
when
Instanciação de componentes
Declaração de processos

Declarações
sequenciais

Declarações IF-THEN
Declaração CASE
Loop statements
Wait statements
Signal attributes

VHDL and Logic
Synthesis

Máquinas de Estados
Finitas (FSMs)

Bibliografia

Equivalent Functions?? NO

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
```

```
ENTITY simp IS
  PORT (
    a, b : IN STD_LOGIC;
    y : OUT STD_LOGIC
  );
END ENTITY simp;
```

```
ARCHITECTURE logic OF simp IS
  SIGNAL c : STD_LOGIC;
  BEGIN
    c <= a AND b;
    y <= c;
  END ARCHITECTURE logic;
```

New value of **c** not available for **y** until
next process execution (requires another
simulation cycle or transition on **a/b**)



```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
```

```
ENTITY simp_prc IS
  PORT (
    a, b : IN STD_LOGIC;
    y: OUT STD_LOGIC
  );
END ENTITY simp_prc;
```

```
ARCHITECTURE logic OF simp_prc IS
  SIGNAL c: STD_LOGIC;
  BEGIN
    PROCESS (a, b)
      BEGIN
        c <= a AND b;
        y <= c;
      END PROCESS;
    END ARCHITECTURE logic;
```

Declaração de processos

Objetivos

Linguagem de
descrição de
hardware VHDL

Entidade de projeto
Corpo da arquitetura
Classes de objetos
Tipos de dados
Operadores
Sobrecarga de Operadores

Declarções
Concorrentes

select
when
Instanciação de componentes
Declaração de processos

Declarções
sequenciais

Declarções IF-THEN
Declaração CASE
Loop statements
Wait statements
Signal attributes

VHDL and Logic
Synthesis

Máquinas de Estados
Finitas (FSMs)

Bibliografia

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY var IS
    PORT (
        a, b : IN STD_LOGIC;
        y : OUT STD_LOGIC
    );
END ENTITY var;

ARCHITECTURE logic OF var IS
BEGIN
    PROCESS (a, b)
        VARIABLE c : STD_LOGIC;
    BEGIN
        c := a AND b;
        y <= c;
    END PROCESS;
END ARCHITECTURE logic;
```

Variable c updated immediately and new value is available for assigning to y

Variable declaration

Variable assignment

Variable is assigned to a signal to synthesize to a piece of hardware

Declaração de processos

Objetivos

Linguagem de
descrição de
hardware VHDL

Entidade de projeto
Corpo da arquitetura
Classes de objetos
Tipos de dados
Operadores
Sobrecarga de Operadores

Declarações
Concorrentes
select
when
Instanciação de componentes
Declaração de processos

Declarações
sequenciais
Declarações IF-THEN
Declaração CASE
Loop statements
Wait statements
Signal attributes

VHDL and Logic
Synthesis

Máquinas de Estados
Finitas (FSMs)

Bibliografia

ARCHITECTURE

{**SIGNAL** declarations}

label1: **PROCESS**
{**VARIABLE** Declarations}

•
•

label2: **PROCESS**
{**VARIABLE** Declarations}

Declared outside of the
process statements
(Visible to all process
statements)

Declared inside the
PROCESS statements
(locally visible to the
process statements)

Signals vs Variables

Objetivos

Linguagem de
descrição de
hardware VHDL

Entidade de projeto
Corpo da arquitetura
Classes de objetos
Tipos de dados
Operadores
Sobrecarga de Operadores

Declarações
Concorrentes
select
when
Instanciação de componentes
Declaração de processos

Declarações
sequenciais
Declarações IF-THEN
Declaração CASE
Loop statements
Wait statements
Signal attributes

VHDL and Logic
Synthesis

Máquinas de Estados
Finitas (FSMs)

Bibliografia

	Signals ($<=$)	Variables ($:=$)
Assign	<code>assignee <= assignment</code>	<code>assignee := assignment</code>
Utility	Represent circuit interconnect	Represent local storage
Scope	Architecture scope (communicate between processes within architecture)	Local Scope (inside processes)
Behavior	Updated at end of current delta cycle (new value not immediately available)	Updated immediately

Conteúdo

1 Objetivos

2 Linguagem de descrição de hardware VHDL

- Entidade de projeto
- Corpo da arquitetura
- Classes de objetos
- Tipos de dados
- Operadores
- Sobre carga de Operadores

Objetivos

Linguagem de
descrição de
hardware VHDLEntidade de projeto
Corpo da arquitetura
Classes de objetos
Tipos de dados
Operadores
Sobre carga de OperadoresDeclarações
Concorrentesselect
whenInstanciação de componentes
Declaração de processosDeclarações
sequenciaisDeclarações IF-THEN
Declaração CASE
Loop statements
Wait statements
Signal attributesVHDL and Logic
SynthesisMáquinas de Estados
Finitas (FSMs)

Bibliografia

3 Declarações Concorrentes

- select
- when
- Instanciação de componentes
- Declaração de processos

4 Declarações sequenciais

- Declarações IF-THEN
- Declaração CASE
- Loop statements
- Wait statements
- Signal attributes

5 VHDL and Logic Synthesis

6 Máquinas de Estados Finitas (FSMs)

7 Bibliografia

Declarações sequenciais

Objetivos

Linguagem de
descrição de
hardware VHDL

Entidade de projeto
Corpo da arquitetura
Classes de objetos
Tipos de dados
Operadores
Sobrecarga de Operadores

Declarações Concorrentes

select
when
Instanciação de componentes
Declaração de processos

Declarações sequenciais

Declarações IF-THEN
Declaração CASE
Loop statements
Wait statements
Signal attributes

VHDL and Logic
Synthesis

Máquinas de Estados
Finitas (FSMs)

Bibliografia

Declarações sequenciais devem estar dentro de processos.

- IF-THEN
- CASE
- Looping statements
- WAIT

Objetivos

Linguagem de
descrição de
hardware VHDLEntidade de projeto
Corpo da arquitetura
Classes de objetos
Tipos de dados
Operadores
Sobrecarga de OperadoresDeclarações
Concorrentes
select
when
Instanciação de componentes
Declaração de processosDeclarações
sequenciais
Declarações IF-THEN
Declaração CASE
Loop statements
Wait statements
Signal attributesVHDL and Logic
SynthesisMáquinas de Estados
Finitas (FSMs)

Bibliografia

Declarações IF-THEN

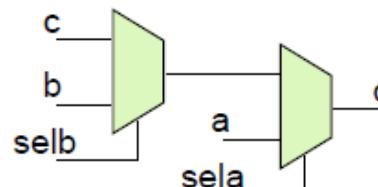
■ Format:

```
IF <condition1> THEN
    {sequence of statement(s)}
ELSIF <condition2> THEN
    {sequence of statement(s)}
    ...
ELSE
    {sequence of statement(s)}
END IF;
```

If-Then statements
act similarly to
conditional signal
assignments

■ Example:

```
PROCESS (sel_a, sel_b, a, b, c)
BEGIN
    IF sel_a='1' THEN
        q <= a;
    ELSIF sel_b='1' THEN
        q <= b;
    ELSE
        q <= c;
    END IF;
END PROCESS;
```



Objetivos

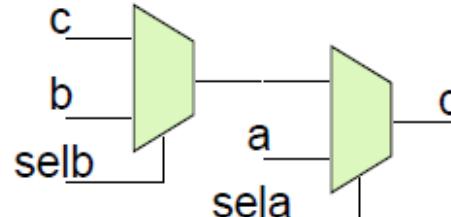
Linguagem de
descrição de
hardware VHDLEntidade de projeto
Corpo da arquitetura
Classes de objetos
Tipos de dados
Operadores
Sobrecarga de OperadoresDeclarações
Concorrentes
select
when
Instanciação de componentes
Declaração de processosDeclarações
sequenciaisDeclarações IF-THEN
Declaração CASE
Loop statements
Wait statements
Signal attributesVHDL and Logic
SynthesisMáquinas de Estados
Finitas (FSMs)

Bibliografia

Declarações IF-THEN

Implicit Process

```
q <= a WHEN sela = '1' ELSE
      b WHEN selb = '1' ELSE
      c;
```

Explicit Process

```
PROCESS (sela, selb, a, b, c)
BEGIN
  IF sela='1' THEN
    q <= a;
  ELSIF selb='1' THEN
    q <= b;
  ELSE
    q <= c;
  END IF;
END PROCESS;
```

Declarações IF-THEN

Objetivos

Linguagem de
descrição de
hardware VHDLEntidade de projeto
Corpo da arquitetura
Classes de objetos
Tipos de dados
Operadores
Sobrecarga de OperadoresDeclarações
Concorrentesselect
whenInstanciação de componentes
Declaração de processosDeclarações
sequenciaisDeclarações IF-THEN
Declaração CASE
Loop statements
Wait statements
Signal attributesVHDL and Logic
SynthesisMáquinas de Estados
Finitas (FSMs)

Bibliografia

A	B	C	F
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

```
entity SystemX is
    port (A, B, C : in bit;
          F        : out bit);
end entity;
```

```
architecture SystemX_arch of SystemX is
begin
    SystemX_Proc : process (A, B, C)
    begin
        if      (A='0' and B='0' and C='0') then F <= '1';
        elsif (A='0' and B='1' and C='0') then F <= '1';
        elsif (A='1' and B='1' and C='0') then F <= '1';
        else F <= '0';
        end if;
    end process;

end architecture;
```

Declaração CASE

Objetivos

Linguagem de
descrição de
hardware VHDL

Entidade de projeto
Corpo da arquitetura
Classes de objetos
Tipos de dados
Operadores
Sobrecarga de Operadores

Declarções
Concorrentes
select
when
Instanciação de componentes
Declaração de processos

Declarções sequenciais
Declarações IF-THEN
Declaração CASE
Loop statements
Wait statements
Signal attributes

VHDL and Logic
Synthesis

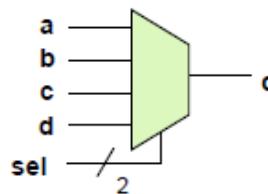
Máquinas de Estados
Finitas (FSMs)

Bibliografia

- Todas as condições são avaliadas ao mesmo tempo.
- Todas as possíveis condições devem ser consideradas.
- “WHEN OTHERS” avalia todas as outras condições possíveis que não são especificamente declaradas.

Format:

```
CASE {expression} IS
    WHEN <condition1> =>
        {sequence of statements}
    WHEN <condition2> =>
        {sequence of statements}
    ...
    WHEN OTHERS => -- (optional)
        {sequence of statements}
END CASE;
```



Example:

```
PROCESS (sel, a, b, c, d)
BEGIN
    CASE sel IS
        WHEN "00" =>
            q <= a;
        WHEN "01" =>
            q <= b;
        WHEN "10" =>
            q <= c;
        WHEN OTHERS =>
            q <= d;
    END CASE;
END PROCESS;
```

Objetivos

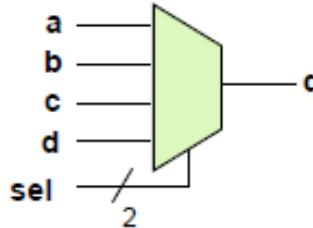
Linguagem de
descrição de
hardware VHDLEntidade de projeto
Corpo da arquitetura
Classes de objetos
Tipos de dados
Operadores
Sobrecarga de OperadoresDeclarações
Concorrentes
select
when
Instanciação de componentes
Declaração de processosDeclarações
sequenciais
Declaração IF-THEN
Declaração CASE
Loop statements
Wait statements
Signal attributesVHDL and Logic
SynthesisMáquinas de Estados
Finitas (FSMs)

Bibliografia

Declaração CASE

Implicit Process

```
WITH sel SELECT
    q <= a WHEN "00",
            b WHEN "01",
            c WHEN "10",
            d WHEN OTHERS;
```

Explicit Process

```
PROCESS (sel, a, b, c, d)
BEGIN
    CASE sel IS
        WHEN "00" =>
            q <= a;
        WHEN "01" =>
            q <= b;
        WHEN "10" =>
            q <= c;
        WHEN OTHERS =>
            q <= d;
    END CASE;
END PROCESS;
```

Declaração CASE

A	B	C	F
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

```
entity SystemX is
    port (A, B, C : in bit;
          F        : out bit);
end entity;
```

```
architecture SystemX_arch of SystemX is
begin
    SystemX_Proc : process (A, B, C)
        variable ABC : bit_vector (2 downto 0) := "000";
        begin
            ABC := A & B & C;
            case (ABC) is
                when "000" => F <= '1';
                when "001" => F <= '0';
                when "010" => F <= '1';
                when "011" => F <= '0';
                when "100" => F <= '0';
                when "101" => F <= '0';
                when "110" => F <= '1';
                when "111" => F <= '0';
            end case;
        end process;
    end architecture;
```

Objetivos

Linguagem de descrição de hardware VHDL

Entidade de projeto

Corpo da arquitetura

Classes de objetos

Tipos de dados

Operadores

Sobrecarga de Operadores

Declarações

Concorrentes

select

when

Instanciação de componentes

Declaração de processos

Declarações sequenciais

Declarações IF-THEN

Declaração CASE

Loop statements

Wait statements

Signal attributes

VHDL and Logic

Synthesis

Máquinas de Estados Finitas (FSMs)

Bibliografia

Loop statements

- While loops
- for loops

While Loop

```
WHILE <condition> LOOP
    --Sequential statements
END LOOP;
```

For Loop

```
FOR <identifier> IN <range> LOOP
    --Sequential statements
END LOOP;
```

Loop statements

Objetivos

Linguagem de
descrição de
hardware VHDL

Entidade de projeto
Corpo da arquitetura
Classes de objetos
Tipos de dados
Operadores
Sobrecarga de Operadores

Declarações
Concorrentes

select
when
Instanciação de componentes
Declaração de processos

Declarações
sequenciais

Declarações IF-THEN
Declaração CASE
Loop statements
Wait statements
Signal attributes

VHDL and Logic
Synthesis

Máquinas de Estados
Finitas (FSMs)

Bibliografia

```

LIBRARY IEEE;
USE IEEE. STD_LOGIC_1164.ALL;
USE IEEE. STD_LOGIC_UNSIGNED.ALL ;
USE IEEE. STD_LOGIC_ARITH.ALL ;

ENTITY ones_count IS
    PORT (
        invec : IN STD_LOGIC_VECTOR (31 DOWNTO 0);
        outvec : OUT STD_LOGIC_VECTOR (7 DOWNTO 0)
    );
END ENTITY ones_count;

ARCHITECTURE rtl OF ones_count IS
BEGIN
    PROCESS (invec)
        VARIABLE count: STD_LOGIC_VECTOR (7 DOWNTO 0);
    BEGIN
        count:= (OTHERS=>'0');
        FOR i IN invec RANGE LOOP
            IF (invec(i) /= '0') THEN
                count:=count+1;
            END IF;
        END LOOP;
        outvec <=count;
    END PROCESS;
END ARCHITECTURE rtl;

```

Variable **count** is initialized

i is the loop index.
This loop examines all 32 bits of **invec**. If the current bit does not equal zero, **count** is incremented.

Variable **count** is assigned to a signal (**outvec**) before the end of the process to be visible outside process & to synthesize to a piece of hardware

Wait statements

Objetivos

Linguagem de
descrição de
hardware VHDL

Entidade de projeto
Corpo da arquitetura
Classes de objetos
Tipos de dados
Operadores
Sobrecarga de Operadores

Declarações
Concorrentes

select
when
Instanciação de componentes
Declaração de processos

Declarações
sequenciais

Declarações IF-THEN
Declaração CASE
Loop statements
Wait statements
Signal attributes

VHDL and Logic
Synthesis

Máquinas de Estados
Finitas (FSMs)

Bibliografia

- Pausa a execução do processo enquanto a declaração WAIT é satisfeita.

- Tipos:

- ▶ WAIT ON <signal>
- ▶ WAIT UNTIL <boolean expression>
- ▶ WAIT FOR <time expression>
- ▶ WAIT UNTIL (a='1') FOR 5 us

```
stim: PROCESS
  VARIABLE error : BOOLEAN;
BEGIN
  WAIT UNTIL clk = '0';
  a <= (OTHERS => '0');
  b <= (OTHERS => '0');
  WAIT FOR 40 NS;
  IF (sum /= 0) THEN
    error := TRUE;
  END IF;

  WAIT UNTIL clk = '0';
  a <= "0010";
  b <= "0011";
  WAIT FOR 40 NS;
  IF (sum /= 5) THEN
    error := TRUE;
  END IF;

  ...
  WAIT;
END PROCESS stim;
```

The diagram illustrates three types of wait statements annotated in the VHDL code:

- Annotation 1:** Points to the first `WAIT UNTIL` statement. The annotation text is: "Pause execution of the process until clk transitions to a logic 0".
- Annotation 2:** Points to the second `WAIT FOR` statement. The annotation text is: "Pause execution of the process until the equivalent of 40 ns passes in simulation time".
- Annotation 3:** Points to the final `WAIT` statement. The annotation text is: "Pause execution of the process indefinitely".

Signal attributes

Objetivos

Linguagem de
descrição de
hardware VHDL

Entidade de projeto

Corpo da arquitetura

Classe de objetos

Tipos de dados

Operadores

Sobrecarga de Operadores

Declarações

Concorrentes

select

when

Instanciação de componentes

Declaração de processos

Declarações
sequenciais

Declarações IF-THEN

Declaração CASE

Loop statements

Wait statements

Signal attributes

VHDL and Logic

Synthesis

Máquinas de Estados

Finitas (FSMs)

Bibliografia

Há situações em que desejamos descrever um comportamento que é baseado em mais do que apenas o valor atual de um sinal.

Attribute	Information returned	Type returned
A'event	True when signal A changes, false otherwise	Boolean
A'active	True when an assignment is made to A, false otherwise	Boolean
A'last_event	Time when signal A last changed	Time
A'last_active	Time when signal A was last assigned to	Time
A'last_value	The previous value of A	Same type as A
B'length	Size of the vector (e.g., 8)	Integer
B'left	Left bound of the vector (e.g., 7)	Integer
B'right	Right bound of the vector (e.g., 0)	Integer
B'range	Range of the vector "(7 downto 0)"	String

Signal attributes

Objetivos

Linguagem de
descrição de
hardware VHDL

Entidade de projeto

Corpo da arquitetura

Classe de objetos

Tipos de dados

Operadores

Sobrecarga de Operadores

Declarações

Concorrentes

select

when

Instanciação de componentes

Declaração de processos

Declarações

sequenciais

Declarações IF-THEN

Declaração CASE

Loop statements

Wait statements

Signal attributes

VHDL and Logic

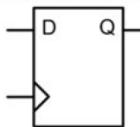
Synthesis

Máquinas de Estados

Finitas (FSMs)

Bibliografia

Example: Behavioral Modeling of a Rising Edge Triggered D-Flip-Flop Using Attributes



Clk	D	Q	
0	X	Last Q	Store
1	X	Last Q	Store
↓	0	0	Update
↓	1	1	Update

```
entity Dflipflop is
  port  (Clock      : in  bit;
         D          : in  bit;
         Q          : out bit);
end entity;

architecture Dflipflop_arch of Dflipflop is
begin
  D_FLIP_FLOP : process (Clock)
  begin
    if (Clock'event and Clock='1') then
      Q <= D;
    end if;
  end process;
end architecture;
```

Conteúdo

1 Objetivos

2 Linguagem de descrição de hardware VHDL

- Entidade de projeto

Corpo da arquitetura

Classes de objetos

Tipos de dados

Operadores

Sobrecarga de Operadores

Declarações

Concorrentes

select

when

Instanciação de componentes

Declaração de processos

Declarações
sequenciais

Declarações IF-THEN

Declaração CASE

Loop statements

Wait statements

Signal attributes

VHDL and Logic
SynthesisMáquinas de Estados
Finitas (FSMs)

Bibliografia

- Entidade de projeto

- Corpo da arquitetura

- Classes de objetos

- Tipos de dados

- Operadores

- Sobrecarga de Operadores

3 Declarações Concorrentes

- select

- when

- Instanciação de componentes

- Declaração de processos

4 Declarações sequenciais

- Declarações IF-THEN

- Declaração CASE

- Loop statements

- Wait statements

- Signal attributes

5 VHDL and Logic Synthesis

6 Máquinas de Estados Finitas (FSMs)

7 Bibliografia

VHDL and Logic Synthesis

Objetivos

Linguagem de
descrição de
hardware VHDL

Entidade de projeto
Corpo da arquitetura
Classes de objetos
Tipos de dados
Operadores
Sobrecarga de Operadores

Declarações
Concorrentes
select
when
Instanciação de componentes
Declaração de processos

Declarações
sequenciais

Declarações IF-THEN
Declaração CASE
Loop statements
Wait statements
Signal attributes

VHDL and Logic
Synthesis

Máquinas de Estados
Finitas (FSMs)

Bibliografia

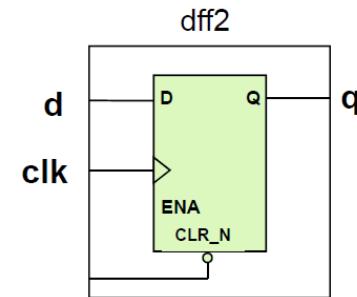
```

LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITYdff2 IS
  PORT (
    d : IN STD_LOGIC;
    clk : IN STD_LOGIC;
    q : OUT STD_LOGIC
  );
END ENTITYdff2;

ARCHITECTURE logic OFdff2 IS
BEGIN
  PROCESS (clk)
  BEGIN
    IF rising_edge (clk) THEN
      q <= d;
    END IF;
  END PROCESS;
END ARCHITECTURE behavior;

```



rising_edge

- IEEE function that is defined in the STD_LOGIC_1164 package
- Specifies that the signal value must be 0 to 1
- X, Z to 1 transition is not allowed

Objetivos

Linguagem de
descrição de
hardware VHDLEntidade de projeto
Corpo da arquitetura
Classes de objetos
Tipos de dados
Operadores
Sobrecarga de OperadoresDeclarações
Concorrentes
select
when
Instanciação de componentes
Declaração de processosDeclarações
sequenciaisDeclarações IF-THEN
Declaração CASE
Loop statements
Wait statements
Signal attributesVHDL and Logic
SynthesisMáquinas de Estados
Finitas (FSMs)

Bibliografia

VHDL and Logic Synthesis

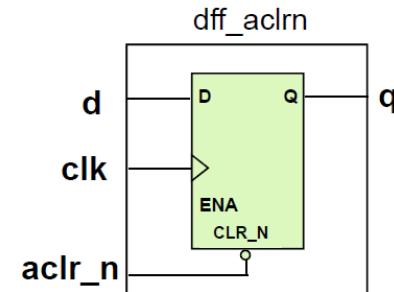
```

LIBRARY IEEE;
USE IEEE. STD_LOGIC_1164.ALL;
USE IEEE. STD_LOGIC_UNSIGNED.All;

ENTITY dff_aclrn IS
    PORT (
        aclr_n : IN STD_LOGIC;
        d, clk : IN STD_LOGIC;
        q : OUT STD_LOGIC
    );
END ENTITY dff_aclrn;

ARCHITECTURE logic OF dff_aclrn IS
BEGIN
    PROCESS (clk, aclr_n)
    BEGIN
        IF aclr_n = '0' THEN
            q <= '0';
        ELSIF rising_edge (clk) THEN
            q <= d;
        END IF;
    END PROCESS;
END ARCHITECTURE behavior;

```



- This is how to implement asynchronous control signals for the register*
- Asynchronous control signal is included in the sensitivity list and is checked before the clock condition (**rising_edge**)*
- Therefore, **aclr_n='0'** does not depend on the clock*

Objetivos

Linguagem de
descrição de
hardware VHDL

Entidade de projeto
Corpo da arquitetura
Classes de objetos
Tipos de dados
Operadores
Sobrecarga de Operadores

Declarações
Concorrentes
select
when
Instanciação de componentes
Declaração de processos

Declarações
sequenciais

Declarações IF-THEN
Declaração CASE
Loop statements
Wait statements
Signal attributes

VHDL and Logic
Synthesis

Máquinas de Estados
Finitas (FSMs)

Bibliografia

VHDL and Logic Synthesis

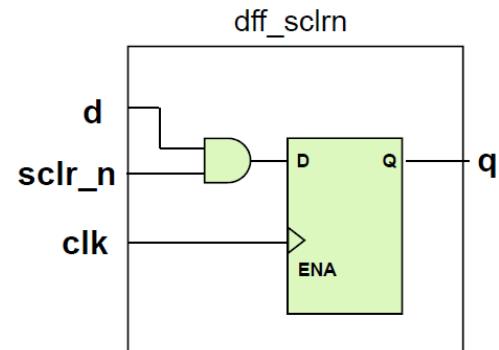
```

LIBRARY IEEE;
USE IEEE. STD_LOGIC_1164.ALL;
USE IEEE. STD_LOGIC_UNSIGNED.ALL;

ENTITY dff_sclrn IS
  PORT (
    sclr_n : IN STD_LOGIC;
    d, clk : IN STD_LOGIC;
    q : OUT STD_LOGIC
  );
END ENTITY dff_sclrn;

ARCHITECTURE logic OF dff_sclrn IS
BEGIN
  PROCESS (clk)
  BEGIN
    IF rising_edge (clk) THEN
      IF sclr_n = '0' THEN
        q <= '0';
      ELSE
        q <= d;
      END IF;
    END IF;
  END PROCESS;
END ARCHITECTURE behavior;

```



- This is how to implement synchronous control signals for the register*
- Synchronous control signals are not included in the sensitivity list and are checked inside in the rising_edge IF-THEN statement that checks the condition **rising_edge***
- Therefore, **sclr_n='0'** does depend on the clock*

Objetivos

Linguagem de
descrição de
hardware VHDL

Entidade de projeto
Corpo da arquitetura
Classes de objetos
Tipos de dados
Operadores
Sobrecarga de Operadores

Declarações
Concorrentes
select
when
Instanciação de componentes
Declaração de processos

Declarações
sequenciais

Declarações IF-THEN
Declaração CASE
Loop statements
Wait statements
Signal attributes

VHDL and Logic
Synthesis

Máquinas de Estados
Finitas (FSMs)

Bibliografia

VHDL and Logic Synthesis

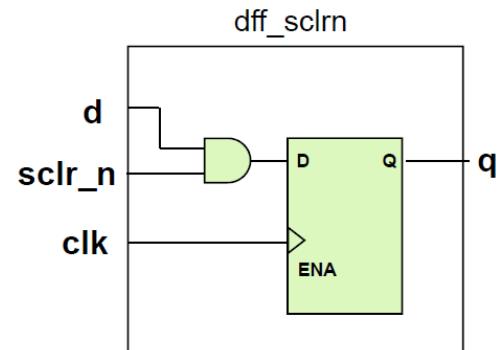
```

LIBRARY IEEE;
USE IEEE. STD_LOGIC_1164.ALL;
USE IEEE. STD_LOGIC_UNSIGNED.ALL;

ENTITY dff_sclrn IS
  PORT (
    sclr_n : IN STD_LOGIC;
    d, clk : IN STD_LOGIC;
    q : OUT STD_LOGIC
  );
END ENTITY dff_sclrn;

ARCHITECTURE logic OF dff_sclrn IS
BEGIN
  PROCESS (clk)
  BEGIN
    IF rising_edge (clk) THEN
      IF sclr_n = '0' THEN
        q <= '0';
      ELSE
        q <= d;
      END IF;
    END IF;
  END PROCESS;
END ARCHITECTURE behavior;

```



- This is how to implement synchronous control signals for the register*
- Synchronous control signals are not included in the sensitivity list and are checked inside in the rising_edge IF-THEN statement that checks the condition **rising_edge***
- Therefore, **sclr_n='0'** does depend on the clock*

Quantos registradores?

VHDL and Logic Synthesis

Objetivos

Linguagem de descrição de hardware VHDL
Entidade de projeto

Corpo da arquitetura
Classes de objetos
Tipos de dados
Operadores
Sobrecarga de Operadores

Declarções
Concorrentes
select
when
Instanciação de componentes
Declaração de processos

Declarções sequenciais
Declarções IF-THEN
Declaração CASE
Loop statements
Wait statements
Signal attributes

VHDL and Logic
Synthesis

Máquinas de Estados
Finitas (FSMs)

Bibliografia

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY reg1 IS
    PORT (
        d : IN STD_LOGIC;
        clk : IN STD_LOGIC;
        q : OUT STD_LOGIC
    );
END ENTITY reg1;

ARCHITECTURE logic OF reg1 IS
    SIGNAL a, b : STD_LOGIC;
BEGIN
    PROCESS (clk)
    BEGIN
        IF rising_edge (clk) THEN
            a <= d;
            b <= a;
            q <= b;
        END IF;
    END PROCESS;
END ARCHITECTURE reg1;
```

VHDL and Logic Synthesis

Objetivos

Linguagem de
descrição de
hardware VHDL

Entidade de projeto
Corpo da arquitetura
Classes de objetos
Tipos de dados
Operadores
Sobrecarga de Operadores

Declarações

Concorrentes

select
when

Instanciação de componentes
Declaração de processos

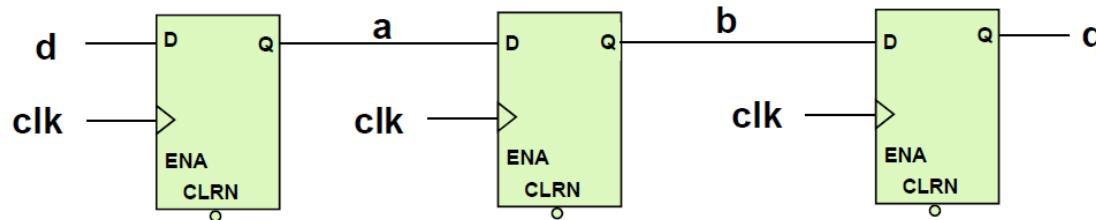
Declarações sequenciais

Declarações IF-THEN
Declaração CASE
Loop statements
Wait statements
Signal attributes

VHDL and Logic Synthesis

Máquinas de Estados
Finitas (FSMs)

Bibliografia



VHDL and Logic Synthesis

Objetivos

Linguagem de
descrição de
hardware VHDL

Entidade de projeto
Corpo da arquitetura
Classes de objetos
Tipos de dados
Operadores
Sobrecarga de Operadores

Declarações
Concorrentes

select
when
Instanciação de componentes
Declaração de processos

Declarações
sequenciais

Declarações IF-THEN
Declaração CASE
Loop statements
Wait statements
Signal attributes

VHDL and Logic
Synthesis

Máquinas de Estados
Finitas (FSMs)

Bibliografia

Quantos registradores?

```

LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY reg3 IS
  PORT (
    d : IN STD_LOGIC;
    clk : IN STD_LOGIC;
    q : OUT STD_LOGIC
  );
END ENTITY reg3;

ARCHITECTURE logic OF reg3 IS
BEGIN
  PROCESS (clk)
    VARIABLE a, b : STD_LOGIC;
  BEGIN
    IF rising_edge (clk) THEN
      a := d;
      b := a;
      q <= b;
    END IF;
  END PROCESS;
END ARCHITECTURE reg1;

```

Signals changed to variables

VHDL and Logic Synthesis

Objetivos

Linguagem de
descrição de
hardware VHDL

- Entidade de projeto
- Corpo da arquitetura
- Clases de objetos
- Tipos de dados
- Operadores
- Sobrecarga de Operadores

Declarações
Concorrentes

- select
- when
- Instanciação de componentes
- Declaração de processos

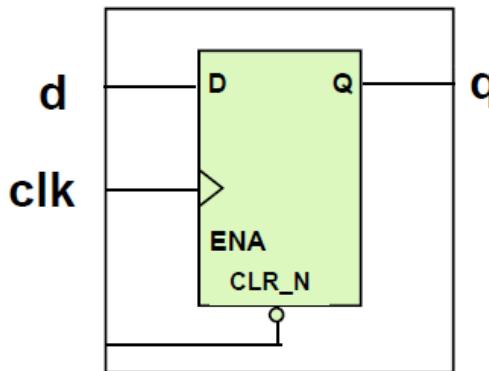
Declarações
sequenciais

- Declarações IF-THEN
- Declaração CASE
- Loop statements
- Wait statements
- Signal attributes

VHDL and Logic
Synthesis

Máquinas de Estados
Finitas (FSMs)

Bibliografia



Objetivos

Linguagem de
descrição de
hardware VHDLEntidade de projeto
Corpo da arquitetura
Classes de objetos
Tipos de dados
Operadores
Sobrecarga de OperadoresDeclarações
Concorrentesselect
whenInstanciação de componentes
Declaração de processosDeclarações
sequenciaisDeclarações IF-THEN
Declaração CASE
Loop statements
Wait statements
Signal attributesVHDL and Logic
SynthesisMáquinas de Estados
Finitas (FSMs)

Bibliografia

```

LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;

ENTITY counter IS
  PORT (
    clk, aclr, clk_ena : IN STD_LOGIC;
    q : OUT STD_LOGIC_VECTOR (15 DOWNTO 0)
  );
END ENTITY counter;

ARCHITECTURE logic OF counter IS
BEGIN
  PROCESS (clk)
    VARIABLE q_var : STD_LOGIC_VECTOR (15 DOWNTO 0);
  BEGIN
    IF aclr = '1' THEN
      q_var := (OTHERS => '0');
    ELSIF rising_edge (clk) THEN
      IF clk_ena = '1' THEN
        q_var := q_var + 1;
      END IF;
    END IF;
    q <= q_var;
  END PROCESS;
END ARCHITECTURE logic;

```

- Counters are accumulators that always add a '1' or subtract a '1'

Arithmetic expression assigned to variable before writing known value

Variable assigned to a signal to create hardware

Conteúdo

1 Objetivos

2 Linguagem de descrição de hardware VHDL

- Entidade de projeto
- Corpo da arquitetura
- Classes de objetos
- Tipos de dados
- Operadores
- Sobre carga de Operadores

Objetivos

Linguagem de
descrição de
hardware VHDLEntidade de projeto
Corpo da arquitetura
Classes de objetos
Tipos de dados
Operadores
Sobre carga de OperadoresDeclarações
Concorrentesselect
whenInstanciação de componentes
Declaração de processosDeclarações
sequenciaisDeclarações IF-THEN
Declaração CASE
Loop statements
Wait statements
Signal attributesVHDL and Logic
SynthesisMáquinas de Estados
Finitas (FSMs)

Bibliografia

3 Declarações Concorrentes

- select
- when
- Instanciação de componentes
- Declaração de processos

4 Declarações sequenciais

- Declarações IF-THEN
- Declaração CASE
- Loop statements
- Wait statements
- Signal attributes

5 VHDL and Logic Synthesis

6 Máquinas de Estados Finitas (FSMs)

7 Bibliografia

Máquinas de Estados Finitos (FSMs)

Objetivos

Linguagem de
descrição de
hardware VHDL

Entidade de projeto
Corpo da arquitetura
Classes de objetos
Tipos de dados
Operadores
Sobrecarga de Operadores

Declarações
Concorrentes

select
when

Instanciação de componentes
Declaração de processos

Declarações
sequenciais

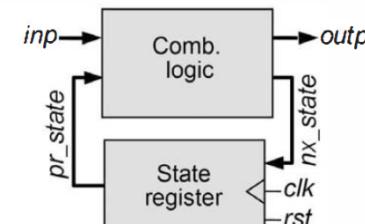
Declarações IF-THEN
Declaração CASE
Loop statements
Wait statements
Signal attributes

VHDL and Logic
Synthesis

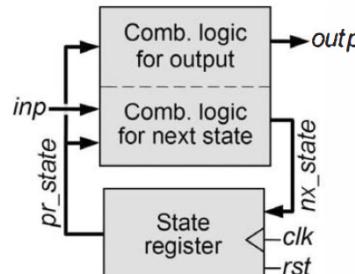
Máquinas de Estados
Finitos (FSMs)

Bibliografia

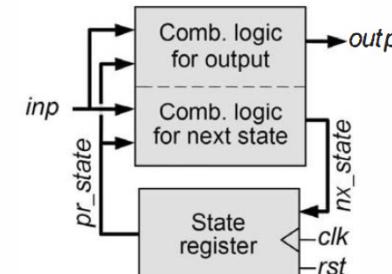
- Utilizar um processo combinacional com uma declaração CASE.
- Utilizar atribuições de sinal condicionais e/ou selecionadas para cada saída.



(a) General diagram



(b) Moore machine



(c) Mealy machine

Máquinas de Estados Finitos (FSMs)

Objetivos

Linguagem de
descrição de
hardware VHDL

Entidade de projeto
Corpo da arquitetura
Classes de objetos
Tipos de dados
Operadores
Sobrecarga de Operadores

Declarações
Concorrentes

select
when
Instanciação de componentes
Declaração de processos

Declarações
sequenciais

Declarações IF-THEN
Declaração CASE
Loop statements
Wait statements
Signal attributes

VHDL and Logic
Synthesis

Máquinas de Estados
Finitas (FSMs)

Bibliografia

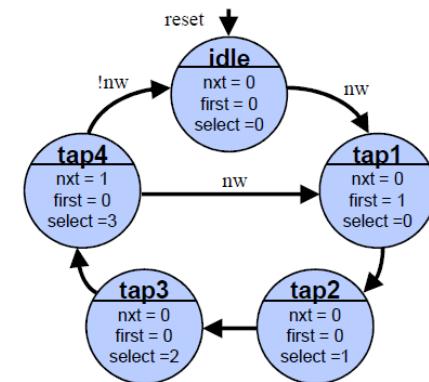
```
LIBRARY IEEE;
USE IEEE. STD_LOGIC_1164.ALL;

ENTITY filter_sm IS
    PORT (
        clk, reset, nw : IN STD_LOGIC;
        select : OUT STD_LOGIC_VECTOR (1 DOWNTO 0);
        nxt, first : OUT STD_LOGIC
    );
END ENTITY filter_sm;
```

```
ARCHITECTURE logic OF filter_sm IS
```

```
TYPE state_type IS (idle, tap1, tap2, tap3, tap4);
SIGNAL filter : state_type;
```

```
BEGIN
```



Enumerated data type

Máquinas de Estados Finitos (FSMs)

Objetivos

Linguagem de
descrição de
hardware VHDL

Entidade de projeto
Corpo da arquitetura
Classes de objetos
Tipos de dados
Operadores
Sobrecarga de Operadores

Declarações
Concorrentes

select
when
Instanciação de componentes
Declaração de processos

Declarações
sequenciais

Declarações IF-THEN
Declaração CASE
Loop statements
Wait statements
Signal attributes

VHDL and Logic
Synthesis

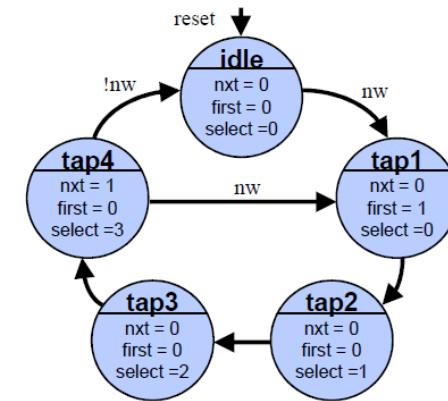
Máquinas de Estados
Finitos (FSMs)

Bibliografia

```

PROCESS (reset, clk)
BEGIN
    IF reset = '1' THEN
        filter <= idle;
    ELSIF clk' EVENT AND clk = '1' THEN
        CASE filter IS
            WHEN idle =>
                IF nw = '1' THEN
                    filter <= tap1;
                END IF;
            WHEN tap1 =>
                filter <= tap2;
            WHEN tap2 =>
                filter <= tap3;
            WHEN tap3 =>
                filter <= tap4;
            WHEN tap4 =>
                IF nw = '1' THEN
                    filter <= tap1;
                ELSE
                    filter <= idle;
                END IF;
        END CASE;
    END IF;
END PROCESS;

```



Máquinas de Estados Finitos (FSMs)

Objetivos

Linguagem de
descrição de
hardware VHDL

Entidade de projeto
Corpo da arquitetura
Classes de objetos
Tipos de dados
Operadores
Sobrecarga de Operadores

Declarações
Concorrentes

select
when
Instanciação de componentes
Declaração de processos

Declarações
sequenciais

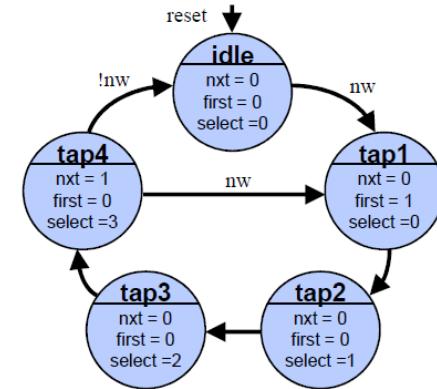
Declarações IF-THEN
Declaração CASE
Loop statements
Wait statements
Signal attributes

VHDL and Logic
Synthesis

Máquinas de Estados
Finitas (FSMs)

Bibliografia

```
output: PROCESS (filter)
BEGIN
    nxt <= '0';
    first <= '0';
    select <= "00";
    CASE filter IS
        WHEN idle =>
        WHEN tap1 =>
            first <= '1';
        WHEN tap2 =>
            select <= "01";
        WHEN tap3 =>
            select <= "10";
        WHEN tap4 =>
            select <= "11";
            nxt <= '1';
    END CASE;
    END PROCESS output;
END ARCHITECTURE logic;
```



Conteúdo

1 Objetivos

2 Linguagem de descrição de hardware VHDL

- Entidade de projeto
- Corpo da arquitetura
- Classes de objetos
- Tipos de dados
- Operadores
- Sobre carga de Operadores

Objetivos

Linguagem de
descrição de
hardware VHDLEntidade de projeto
Corpo da arquitetura
Classes de objetos
Tipos de dados
Operadores
Sobre carga de OperadoresDeclarações
Concorrentesselect
whenInstanciação de componentes
Declaração de processosDeclarações
sequenciaisDeclarações IF-THEN
Declaração CASE
Loop statements
Wait statements
Signal attributesVHDL and Logic
SynthesisMáquinas de Estados
Finitas (FSMs)

Bibliografia

3 Declarações Concorrentes

- select
- when
- Instanciação de componentes
- Declaração de processos

4 Declarações sequenciais

- Declarações IF-THEN
- Declaração CASE
- Loop statements
- Wait statements
- Signal attributes

5 VHDL and Logic Synthesis

6 Máquinas de Estados Finitas (FSMs)

7 Bibliografia

Bibliografia

- Pedroni, V.A. *Circuit Design with VHDL*. MIT Press, 2004
- LaMeres, B. J. (2023). *Introduction to Logic Circuits & Logic Design with VHDL*. Suíça: Springer International Publishing.

Objetivos

Linguagem de
descrição de
hardware VHDL

Entidade de projeto
Corpo da arquitetura
Classe de objetos
Tipos de dados
Operadores
Sobrecarga de Operadores

Declarações Concorrentes

select
when
Instanciação de componentes
Declaração de processos

Declarações sequenciais

Declarações IF-THEN
Declaração CASE
Loop statements
Wait statements
Signal attributes

VHDL and Logic Synthesis

Máquinas de Estados
Finitas (FSMs)

Bibliografia